A TWO-DIRECTIONAL TARGET OPTIMIZATION MODEL

Gregory R. Hamelin

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A TWO-DIRECTIONAL TARGET OPTIMIZATION MODEL

by

Gregory R. Hamelin

March 1979

Thesis Advisor: Gilbert T. Howard

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Two-Directional Target Optimization Model		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1979 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gregory R. Hamelin		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1979 13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents an algorithm for computing the optimal target path for two aircraft traversing a target area from different directions. There are constraints on the maneuverability of each aircraft which prohibit it from attacking every target. The algorithm chooses a subset of targets whose destruction will yield maximum value to the attacking force.

The basis of the algorithm is the branch and bound method, with upper bounds computed by dynamic programming. Several variations are considered, such as payload limit, an increased number of aircraft from each direction, and a three-directional attack. An example problem is solved using the basic model.

A Fortran IV computer program is included. Computation time versus problem characteristics is discussed.

Approved for public release; distribution unlimited.

A TWO-DIRECTIONAL TARGET OPTIMIZATION MODEL

by

Gregory R. Hamelin
Lieutenant, United States Navy
B.S., United States Naval Academy, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1979

ABSTRACT

This paper presents an algorithm for computing the optimal target path for two aircraft traversing a target area from different directions. There are constraints on the maneuverability of each aircraft which prohibit it from attacking every target. The algorithm chooses a subset of targets whose destruction will yield maximum value to the attacking force. The basis of the algorithm is the branch and bound method, with upper bounds computed by dynamic programming. Several variations are considered, such as payload limit, an increased number of aircraft from each direction, and a three-directional attack. An example problem is solved using the basic model.

A Fortran IV computer program is included. Computation time versus problem characteristics is discussed.

TABLE OF CONTENTS

I. INTRODUCTION

Optimal assignment of targets to aircraft on a strike mission can greatly increase the effectiveness of an air attack against ground installations. The purpose of this paper is to present an algorithm which will, given the location and relative military/industrial worth of key enemy positions, select a sequenced subset of targets whose destruction will yield maximum value to the attacking force. It assumes that the decision maker has complete knowledge of the targets and their value. A stochastic extension can easily be incorporated when considering hardened targets. This is done by multiplying the probability of killing a particular target by its value.

The initial course of an aircraft approaching a target area is denoted by ϕ. Due to anti-air defenses, there is a limit placed on the maximum number of degrees which an aircraft may deviate from its initial course. This angular deviation is denoted by θ. Its effect is to restrict the aircraft's movement at any point to a cone whose vertex angle, 2θ, is bisected by the aircraft's initial course, ϕ. Thus at any point (x,y) in the target area, the aircraft's course options are between $\phi-\theta$ and $\phi+\theta$. This creates an ordering among the targets which dictates the sequence in which they must be considered. The allowable deviation is pictured in Figure 1.

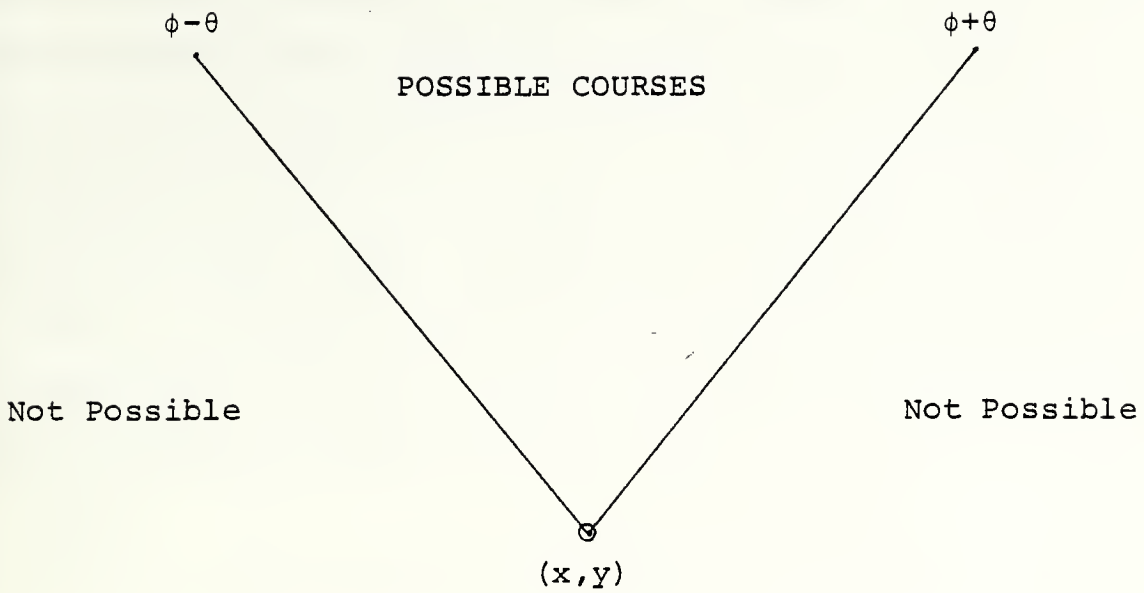


Figure 1. Cone of Allowable Course Deviations

Because of the course constraint, there are only a limited number of paths through the target area which are feasible. The task of the decision maker is to choose that set of targets which yields the optimal combined value.

References [1] and [2] present a method for determining the optimal set of targets for one aircraft traversing an area containing MN targets. The problem is formulated as an MN+1 stage dynamic programming problem [Ref. 3]. The targets are numbered in decreasing order from an imaginary line drawn tangent to the boundary at which the aircraft enters the target area.

The stages are represented by straight lines drawn through the targets, parallel to the boundary tangent line. Stage n corresponds to the parallel line drawn through target n. The starting point of the aircraft, outside the target area, is stage MN+1. It is located so that any target is accessible from stage MN+1. The stage diagram is depicted in Figure 2. Should two targets be equidistant from the boundary tangent line, either one may be assigned the next sequential number, and the problem reduces to MN stages.

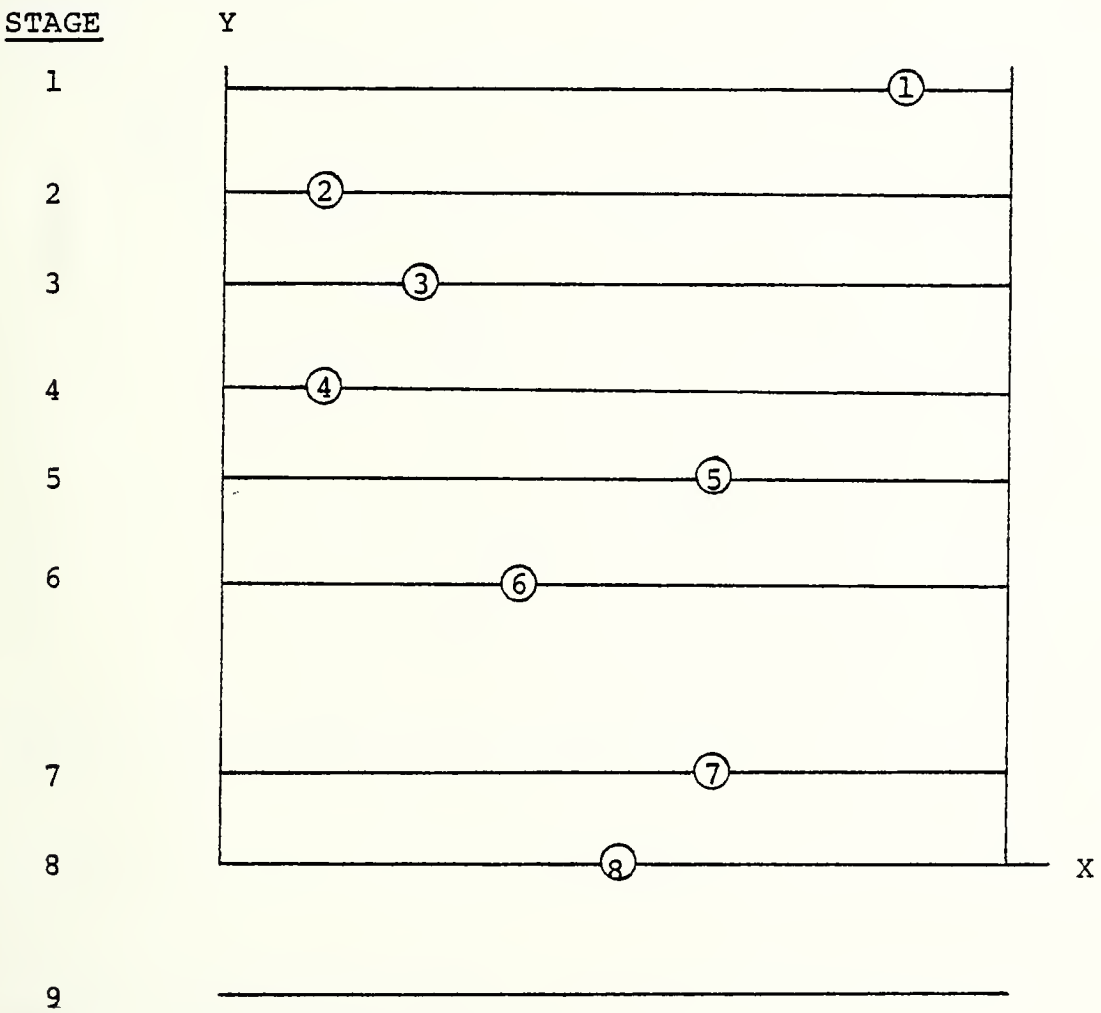


Figure 2. The Stage Diagram for One Aircraft Entering from below the Target Area.

It is important to realize that were the aircraft to enter the target area, say, from the right, the stage diagram would appear as in Figure 3.

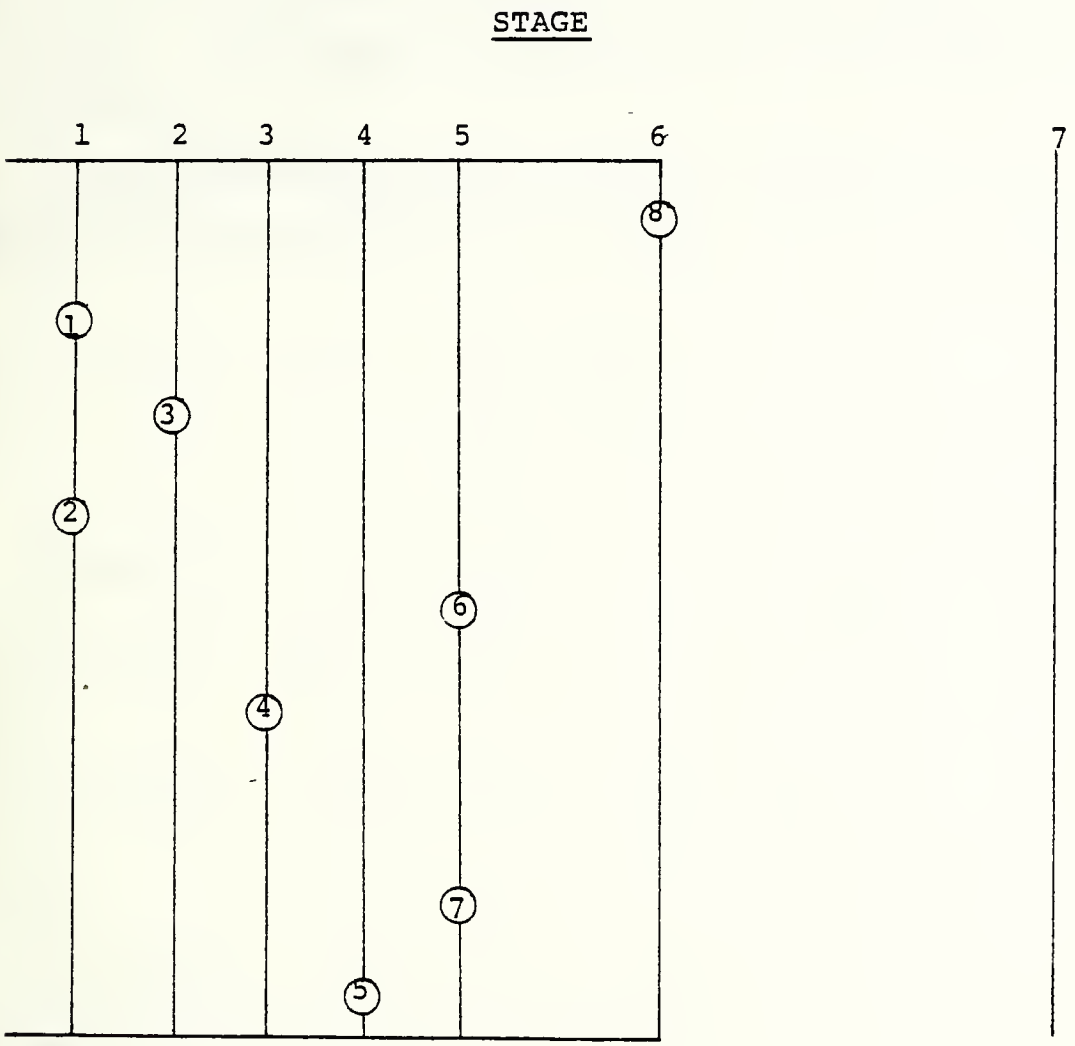


Figure 3. The Stage Diagram for One Aircraft Entering Target Area from the Right.

The state variable x_n denotes the lateral position of the aircraft at stage n. D_n is the decision as to the heading of the aircraft as it moves from stage n to stage n-1. D_n is restricted to lie in the set of feasible headings S_n, from $\phi-\theta$ to $\phi+\theta$. The state variable x_n is then a function of x_{n+1} and D_{n+1}. This function is referred to as the stage transformation t.

The return function for stage n is denoted by r_n. Letting p_n be the lateral position of target n, and V_n be its value,

$$r_n(x_n) = \begin{cases} V_n & \text{if } x_n = p_n \\ 0 & \text{otherwise} \end{cases}$$

The problem is then written as

$$\text{Maximize} \quad \sum_{n=1}^{MN} r_n(x_n)$$

$$\text{Subject to:} \quad x_n = t(x_{n+1}, D_{n+1}) \qquad n=1,\ldots,MN$$

$$D_n \subset S_n \qquad n=2,\ldots,MN+1$$

An efficient algorithm for solving this problem is discussed in Section VI, but the problem can also be easily solved graphically. To do so, the course deviation angle must be viewed from the perspective of the stage diagram. Figure 4 shows whether target n is feasible for various values of x_{n+1}.

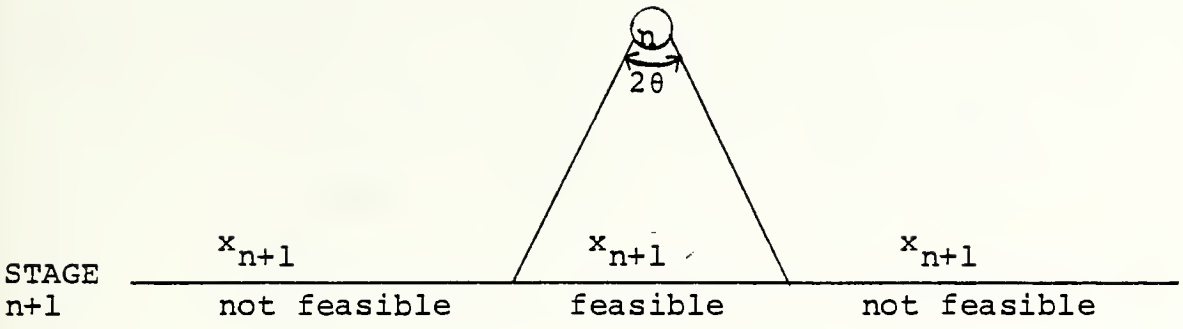


Figure 4. Cone of Feasibility

Using Figure 2 as an example, the maximum return possible for stages (n-1) to 1 is recorded on each stage line n, n = 1, ..., MN+1, for every possible value of x_n. Assuming the value of each target is one, Figure 5 gives the solution for one aircraft entering from below the target area.

Tracing back from stage MN+1, the optimal set of targets is 8, 7, 6, 4, 3, 2, for a value of six.

Dynamic programming can extend this problem to M aircraft by increasing the number of state variables to M [Refs. 1, 2, and 3], as long as all M aircraft are attacking from the same direction. But when the aircraft enter the target area from different directions, a pure dynamic programming solution is no longer possible, since the association between the target and the stage is no longer valid. A target in stage three

for one aircraft may be in stage ten for another, and the recursive equations have no meaning.

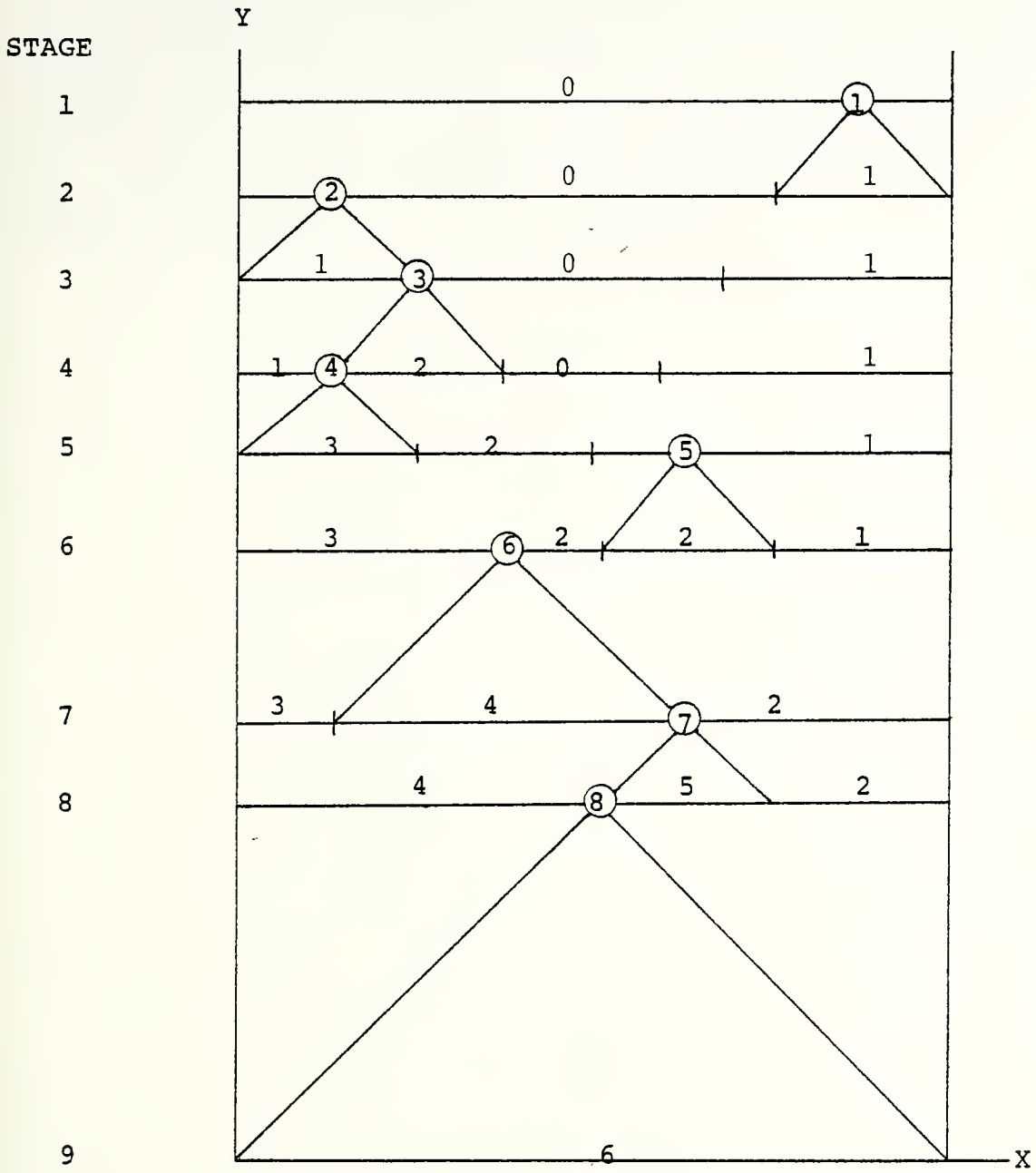


Figure 5. Graphical Solution to the One-Aircraft Problem.

This paper presents a solution to the problem of two aircraft attacking a target area from different directions. A model is formulated, and the algorithm developed. An example problem is included. Also included is a computer program for implementing the algorithm and a discussion of its effectiveness. Because the terminology becomes quite involved, a glossary of terms is provided in Appendix A.

II. PROBLEM FORMULATION

In formulating this problem, it is assumed that if an aircraft attacks a particular target, that target is destroyed. In other words, there are no misses. As was stated in Section I, this does not preclude a stochastic approach for hardened targets where each has a probability of being destroyed.

For ease of notation, the two aircraft are denoted aircraft A and aircraft B.

The targets are numbered as in Section I, but now each target has two numbers, one with respect to aircraft A, and one with respect to aircraft B. They are denoted A_i and B_j, respectively, $i = 1, \ldots, MN$, $j = 1, \ldots, MN$. It is critical to realize that if any target A_i has the same coordinates as any target B_j, then that A_i is the identical target B_j. In fact, for every A_i there will be a B_j identical to it. Figures 2 and 3, which are identical target areas, should clarify this point.

The value of A_i will be denoted by $V[A_i]$, and the value of B_j by $V[B_j]$.

A feasible set of targets for a single aircraft is one in which the course required to go to each successive target is within the allowable course deviation cone described in Section I. PA(I) denotes one of the feasible sets of targets for aircraft A. It is convenient to think of PA(I) as a path of targets which aircraft A will attack. Any PA(I) completely disregards aircraft B in that it is computed as if only

one aircraft were attacking. Similarly, PB(J) denotes one of the feasible sets of targets for aircraft B.

The value of a set of targets, or a path, is the summation of the individual target values comprising that set. The values of PA(I) and PB(J) are denoted by V[PA(I)] and V[PB(J)], respectively.

In order for the solution of a two-aircraft problem to be feasible, the sets of targets for aircraft A and aircraft B must be mutually exclusive, that is, they must have no targets in common. This is so because one aircraft is sufficient to destroy the target. No additional return would be realized by the other aircraft attacking the same target. If this feasibility constraint were not required for optimality, aircraft A (or aircraft B) might forego the opportunity to attack other targets in order to attack the target that both aircraft have in common.

The optimal solution is achieved when PA(I) and PB(J) are chosen so as to

Maximize $V[PA(I)] + V[PB(J)]$

Subject to: $PA(I) \cap PB(J) = \emptyset$.

III. THE ALGORITHM

The branch and bound method forms the basis of the solution algorithm for the two-aircraft problem. It is discussed in theory in Refs. 4, 5, 6, and 7. Only a small fraction of the possible solutions to the problem is actually enumerated. The remaining solutions are eliminated from consideration through the application of bounds that establish that such solutions cannot be optimal.

The algorithm begins by considering all possible combinations of paths for both aircraft. It then breaks this set of all possible combinations into smaller and smaller subsets, and calculates for each an upper bound on the value of the best paths contained therein. The bounds determine the partitioning of the subsets and eventually identify an optimal path for both aircraft. The branch and bound method represents the subsets as nodes of a tree and the partitioning of the subsets as a branching of the tree.

Node one consists of all possible combinations of paths for both aircraft. Using the single aircraft dynamic programming (D.P.) method of Section I, the optimal path for aircraft A is computed for the direction from which A enters the target area. This path is denoted by PA(1). By the same method, the optimal path for aircraft B is computed for the direction from which B enters. This resultant path is denoted by PB(1). It is important to realize that both PA(1) and PB(1) are computed

as single aircraft optimizations, and that all MN targets are possible elements of PB(1), even those in PA(1).

If PA(1) and PB(1) have no targets in common, then the paths form the optimal solution to the two-aircraft problem, the value of which is V[PA(1)] + V[PB(1)], and the algorithm stops. However, if there are targets in common, then as was mentioned in Section II, the solution is not feasible. The summation V[PA(1)] + V[PB(1)] represents instead an upper bound, denoted by UB(1), for the optimal solution. The summation of the values of the points of intersection of PA(1) and PB(1) is denoted by VINT(1). The target in the intersection which has the highest value is denoted by $X_{AB}(1)$. This target could be written in terms of aircraft A or aircraft B. When the distinction is necessary, $X_{AB}(1)$ will be written as either $X_{AB}(1)_A$ or $X_{AB}(1)_B$, for aircraft A and aircraft B, respectively. Geographically, however, $X_{AB}(1)$, $X_{AB}(1)_A$, and $X_{AB}(1)_B$ are identical.

The goal is to find optimal paths for aircraft A and B which have no targets in common. The nature of the D.P. solution for a single aircraft path is such that it seeks out those feasible targets with the highest value. In node one, both A and B sought $X_{AB}(1)$. If the set of path combinations in node one was restricted so that aircraft A had to take $X_{AB}(1)_A$ and aircraft B could not take $X_{AB}(1)_B$, the same target, then that point of intersection would be eliminated. But perhaps the optimal solution requires B, not A, to take $X_{AB}(1)$. It could even require that neither path include $X_{AB}(1)$. Therefore, to include all possibilities, the set of node one is broken

into two subsets, one which requires that the path of aircraft A exclude $X_{AB}(1)$, giving aircraft B the option of taking it or not, depending on the single aircraft D.P. solution for B. The other subset requires that the path of aircraft B exclude $X_{AB}(1)$, giving aircraft A the option. Thus node one branches to form nodes two and three.

The restrictions placed on each aircraft at node I are denoted by R(I). If, for example, $R(I) = A_i, B_j, B_k$, the path of aircraft A would be required to exclude target A_i, and the path of aircraft B would be required to exclude targets B_j and B_k.

The branching, with restrictions, is illustrated in Figure 6.

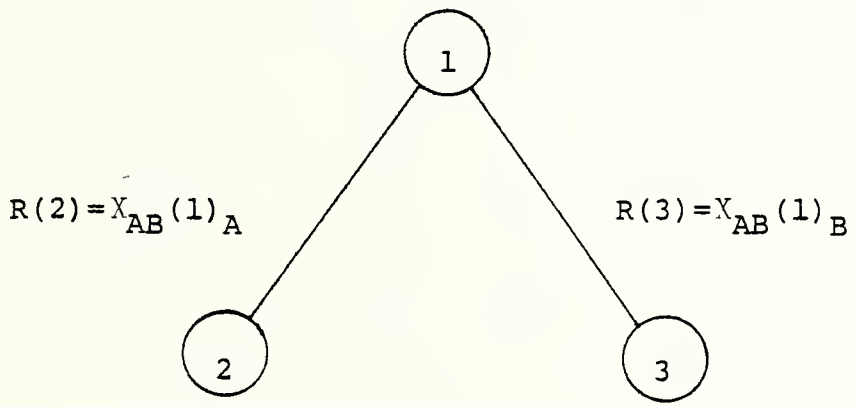


Figure 6. The Start of a Tree

Now consider node two with the restriction vector R(2) containing the single element $X_{AB}(1)_A$. Again using the

single aircraft D.P. method, only this time with the restriction that the path of aircraft A not include target $X_{AB}(1)_A$, the optimal path for aircraft A is computed. The restriction can be incorporated into the solution techniques of Section I by temporarily assigning a large negative value to $X_{AB}(1)$, thereby making it highly unattractive as an element of the optimal path. Denote the resultant path PA(2), and return the original value to $X_{AB}(1)$. As was noted, restricting PA(2) to exclude $X_{AB}(1)_A$ places no restrictions on aircraft B. Therefore, PB(2) will be identical to PB(1). Just as with node one, UB(2), VINT(2), and $X_{AB}(2)$ can be computed.

Node three is considered next, with restriction vector R(3) containing the single element $X_{AB}(1)_B$. Since this means that the path of aircraft B must exclude $X_{AB}(1)_B$, a large negative number is assigned to $X_{AB}(1)_B$, PB(3) is computed, and the original value is returned to $X_{AB}(1)_B$. Since R(3) places no restrictions on PA(3), it is identical to PA(1). Next, UB(3), VINT(3), and $X_{AB}(3)$ are computed.

A terminal node is one from which branching may still occur. Nodes two and three are terminal nodes. Since only two branches may emanate from any node, node one is no longer terminal, and need not be considered further.

The next step in the algorithm is to choose the terminal node which has the highest upper bound. Assume it is node J (at this point, J is either two or three, whichever has the higher upper bound). Node J then branches to form nodes four and five. The restriction vector R(4) will equal R(J) with

the addition of element $X_{AB}(J)_A$. The restriction vector R(5) will equal R(J) with the addition of element $X_{AB}(J)_B$. Again the restricted paths are computed and the algorithm continues until the stopping condition is met.

The branching is always done in pairs, a left branch and a right branch, as was shown in Figure 6. No valid upper bound comparisons can be made until both branchings have been performed and the upper bounds of both new nodes have been computed.

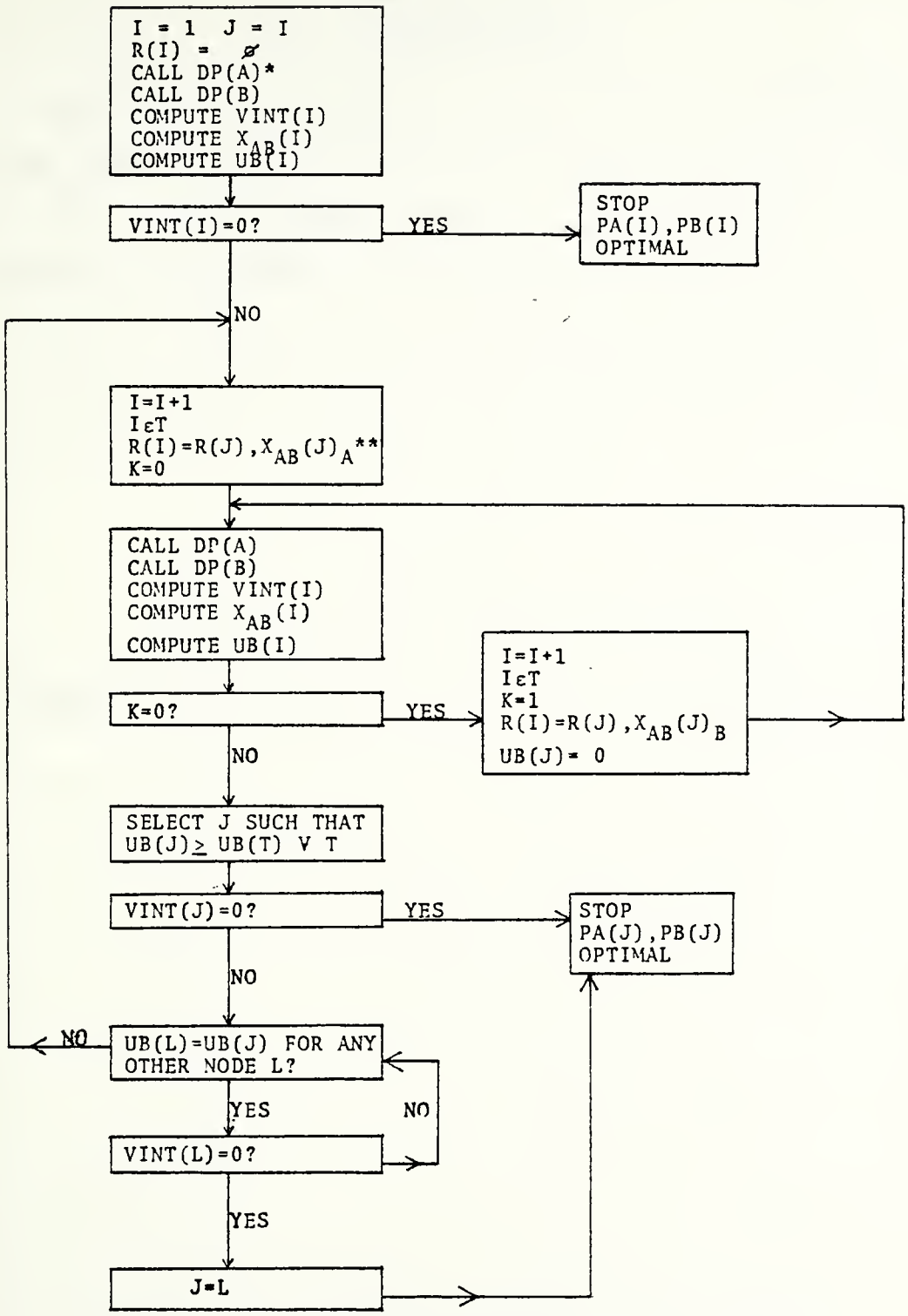
The stopping condition for the algorithm occurs when, following a double branching, a terminal node is found whose upper bound is greater than or equal to all other terminal node upper bounds, and whose paths for aircraft A and aircraft B are mutually exclusive, that is, they have no targets in common. This solution is optimal because its value is equal to the upper bound of that node, making it the best solution possible for that node. And since its value is at least as good as the best solution possible for all other terminal nodes, it is a global optimum.

The steps of the algorithm are summarized in Figure 7 at the end of this section.

There is one other calculation which can be made at each node which is of some interest. The lower bound for each node I is denoted by LB(I) and is equal to UB(I) minus VINT(I). This is a lower bound because if all the points of intersection on PA(I) and PB(I) were given to either aircraft A or aircraft B, the value of the resultant solution would be LB(I).

If any terminal node I has a lower bound on the optimal value which is greater than or equal to the upper bound of any other terminal node J, then node J may be completely dropped from consideration. It is said to be fathomed, and is no longer terminal. This will not speed up the algorithm or reduce the number of branchings required, since the algorithm would never branch from node J anyway. However, if computer storage space were critical, it would be advantageous to incorporate lower bounding, since once a node was fathomed it could be removed from storage. It will not be employed in this algorithm.

The algorithm guarantees that an optimal solution will be found. However, it suffers from a limitation common to all branch and bound methods. For any untried problem, it is impossible to tell beforehand exactly how much computation will actually be necessary to find the optimal solution. Depending on the way the problem is set up, it could converge to the optimum very quickly, or for a large, difficult problem it could require such excessive branching that it becomes computationally prohibitive. This would be the case if the allowable course deviation were very large, the angle between attackers very small, and there were multiple optimal paths. However, in sample problems of target optimization, the algorithm converges very quickly, as will be shown in Section VI.



*CALL DP(A) MEANS TO PERFORM THE SINGLE AIRCRAFT D.P. METHOD USING AIRCRAFT A WITH THE TARGET VALUES ADJUSTED AS REQUIRED BY R(I).

**$R(I)=R(J),X_{AB}(J)_A$ MEANS R(I) INCLUDES THE ELEMENT $X_{AB}(J)_A$ IN ADDITION TO ALL THE ELEMENTS OF R(J).

Figure 7. Flow Chart of the Algorithm.

IV. A TEST PROBLEM

Two aircraft are to attack a target area. Aircraft A has an initial heading of due north, and aircraft B is heading due west as they approach the area. Each aircraft has an allowable course deviation of forty-five degrees.

The target positions and values are given in Table I. The positions are given in terms of the cartesian plane, with the positive Y axis pointing due north.

TABLE I

TARGET POSITIONS AND THEIR VALUES

Target Positions (X, Y)	Target Values
9, 12	1
13, 5	2
4, 0	4
8, 6	2
6, 7	1
11, 14	3
2, 1	2
10, 11	1
5, 10	5
7, 4	1
3, 8	3
1, 13	5
15, 15	2

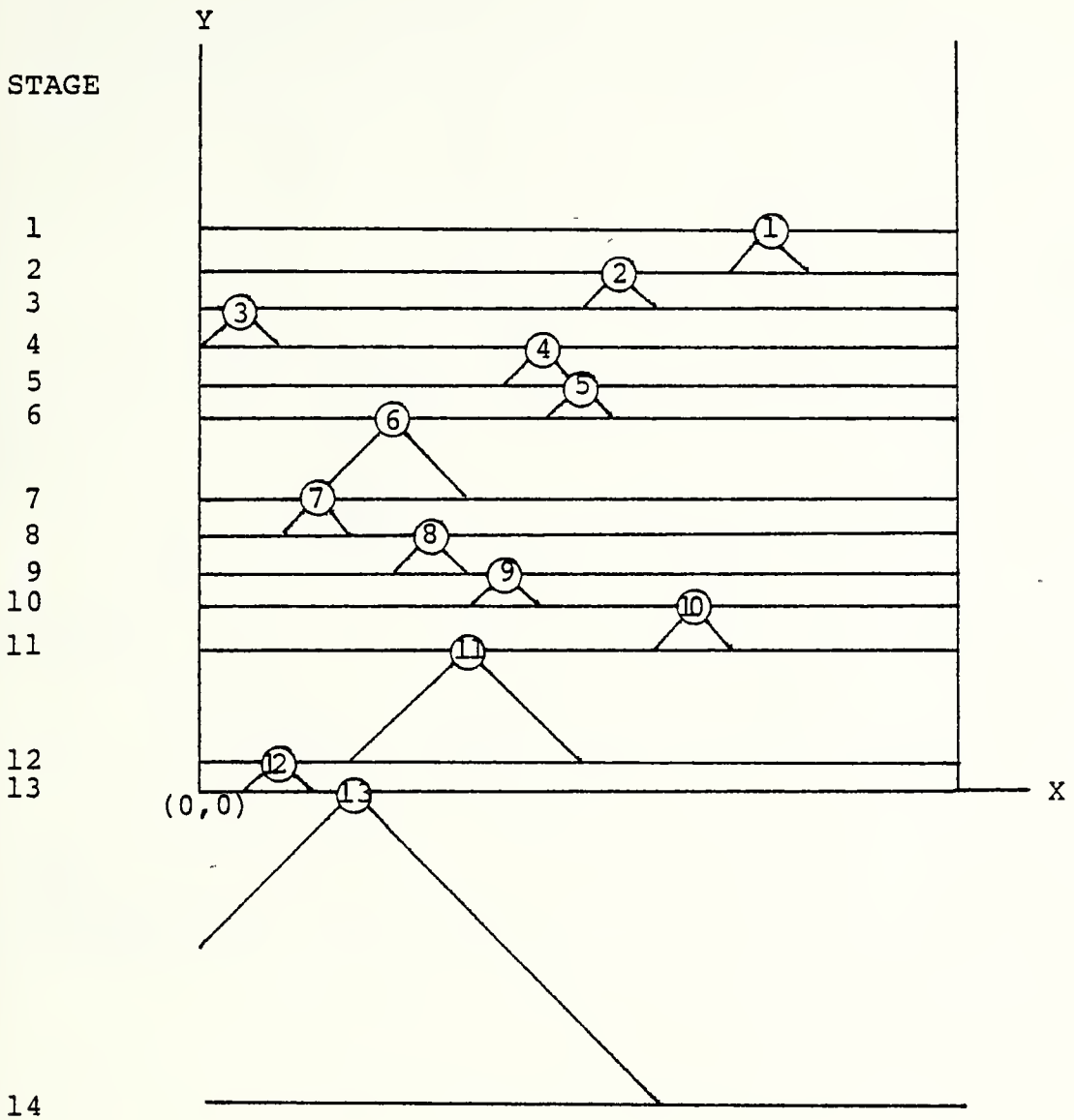


Figure 8. Stage Diagram for Aircraft A

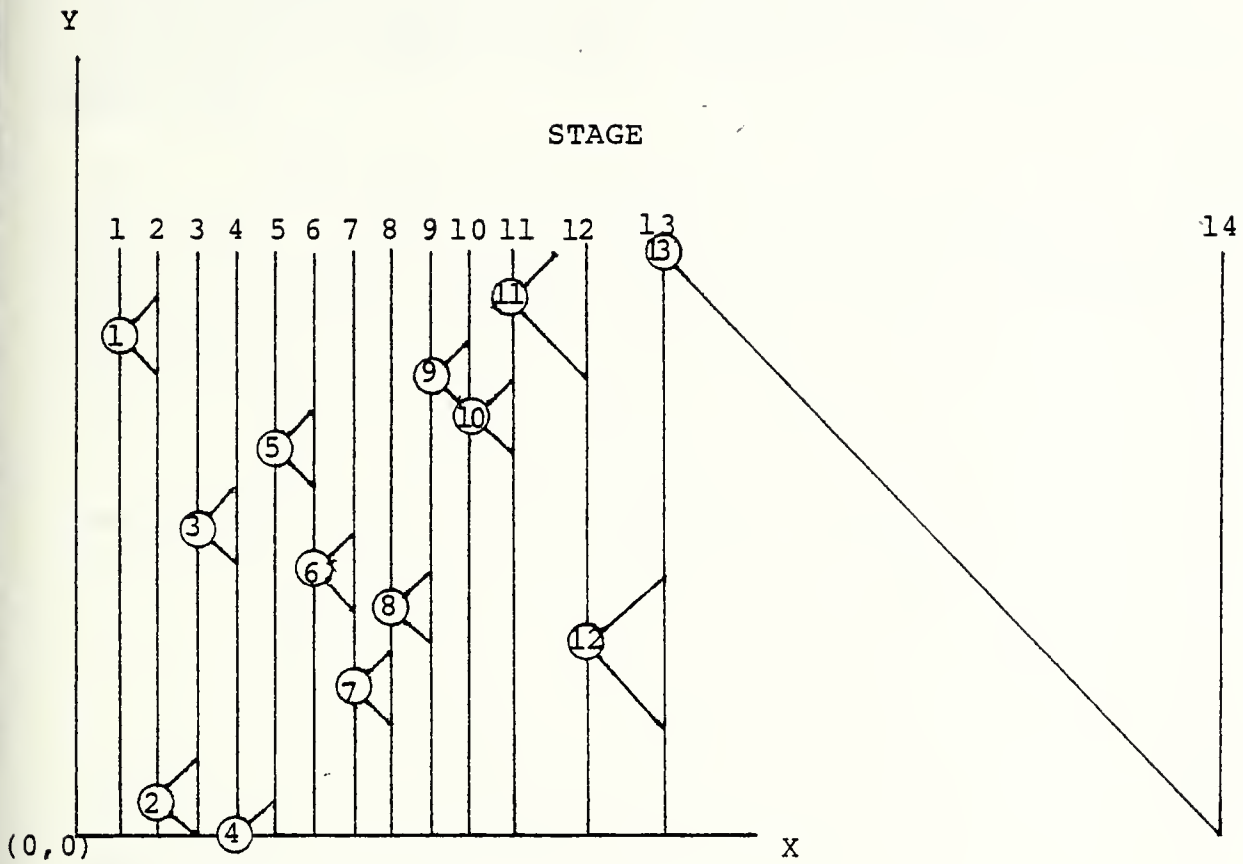


Figure 9. Stage Diagram for Aircraft B

The stage diagrams for aircraft A and aircraft B, including allowable course deviation cones, are illustrated in Figure 8 and Figure 9, respectively.

Beginning with node one, R(1) equals the null set, meaning there are no restrictions on the path of either aircraft. Using the single aircraft D.P. method for aircraft A, it is found that

$$PA(1) = A_{13}, A_{11}, A_7, A_3$$

$PA(1) = A_{13}, A_{11}, A_7, A_6$ is also optimal, and either may be chosen. For this example, the former is used. Similarly, for aircraft B,

$$PB(1) = B_{13}, B_{11}, B_9, B_5, B_1$$

From PA(1) and PB(1), the following values are computed:

$$X_{AB}(1) = A_3 = B_1$$

$$VINT(1) = 5$$

$$UB(1) = V[PA(1)] + V[PB(1)] = 29$$

Since both paths have target $A_3 = B_1$ in common, they do not represent a feasible solution for the two-aircraft problem.

Node one branches to form nodes two and three.

$$R(2) = X_{AB}(1)_A = A_3$$

$$R(3) = X_{AB}(1)_B = B_1$$

The tree at this point is illustrated in Figure 10.

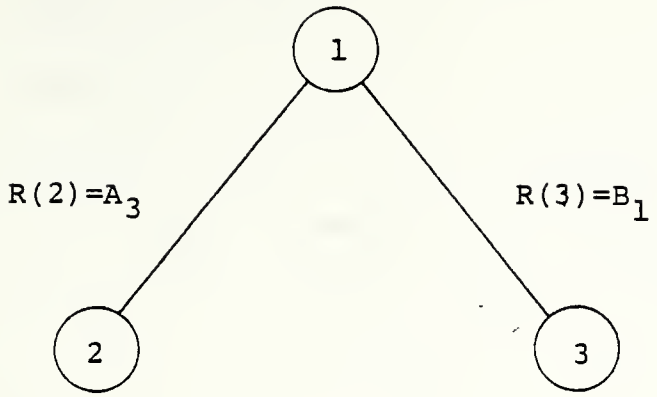


Figure 10. Start of the Tree

Considering node two, PA(2) and PB(2) are computed subject to the restriction that PA(2) may not include target A_3. There are no restrictions on PB(2).

$$PA(2) = A_{13}, A_{11}, A_7, A_6$$

$$PB(2) = B_{13}, B_{11}, B_9, B_5, B_1$$

The target of intersection is $A_6 = B_5$.

$$X_{AB}(2) = A_6 = B_5$$

$$VINT(2) = 5$$

$$UB(2) = 29$$

Moving to node three, PA(3) and PB(3) are computed subject to R(3) which states that PB(3) must exclude target B_1.

$$PA(3) = A_{13}, A_{11}, A_7, A_3$$

$$PB(3) = B_{13}, B_{11}, B_9, B_5, B_3$$

The target of intersection is $A_7 = B_3$.

$X_{AB}(3) = A_7 = B_3$

VINT(3) = 3

UB(3) = 27

Since a double branching has been completed, the upper bounds of all terminal nodes must be compared. The terminal nodes are nodes two and three. Node two has the highest upper bound. Since PA(2) and PB(2) have a target in common, the paths cannot be feasible for the two-aircraft problem. Therefore, node two branches to form nodes four and five. The branching tree expands to Figure 11.

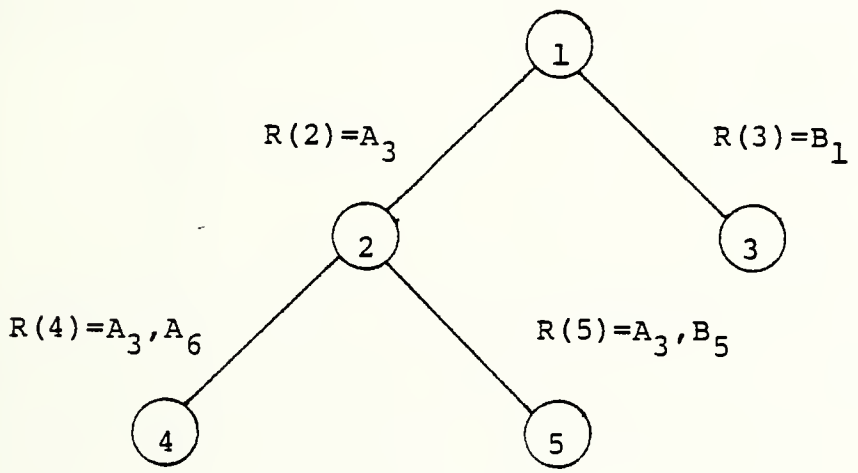


Figure 11. Expanded Branching Tree

Node four is subject to the restrictions from node two in addition to the restriction that PA(4) cannot include A_6.

PA(4) = A_{13}, A_{11}, A_9, A_5, A_4, A_2

PB(4) = B_{13}, B_{11}, B_9, B_5, B_1

Here there are two targets of intersection, $A_4 = B_9$, and $A_2 = B_{11}$.

$$X_{AB}(4) = A_2 = B_{11}$$

$$VINT(4) = V[A_4 = B_9] + V[A_2 = B_{11}] = 4$$

$$UB(4) = 28$$

Node five has R(5) equal to R(2) with the added restriction that PB(5) not include $X_{AB}(2)_B$. Thus R(5) = A_3, B_5. PA(5) must exclude target A_3, and PB(5) must exclude target B_5.

$$PA(5) = A_{13}, A_{11}, A_7, A_6$$

$$PB(5) = B_{13}, B_{11}, B_9, B_1$$

There are no targets of intersection, but this does not mean that the optimal solution has been found, since terminal node upper bounds have not yet been compared.

$$X_{AB}(5) = \emptyset$$

$$VINT(5) = 0$$

$$UB(5) = 24$$

Having completed a double branching, upper bounds are now compared for terminal nodes three, four, and five, and it is found that node four has the highest upper bound. PA(4) and PB(4) have a target in common, and are therefore not feasible for the two-aircraft problem. Since no other terminal node has an upper bound as high as node four, the algorithm branches from node four to form nodes six and seven.

The algorithm continues in this manner with the upper bound computed for each new node, and following each double

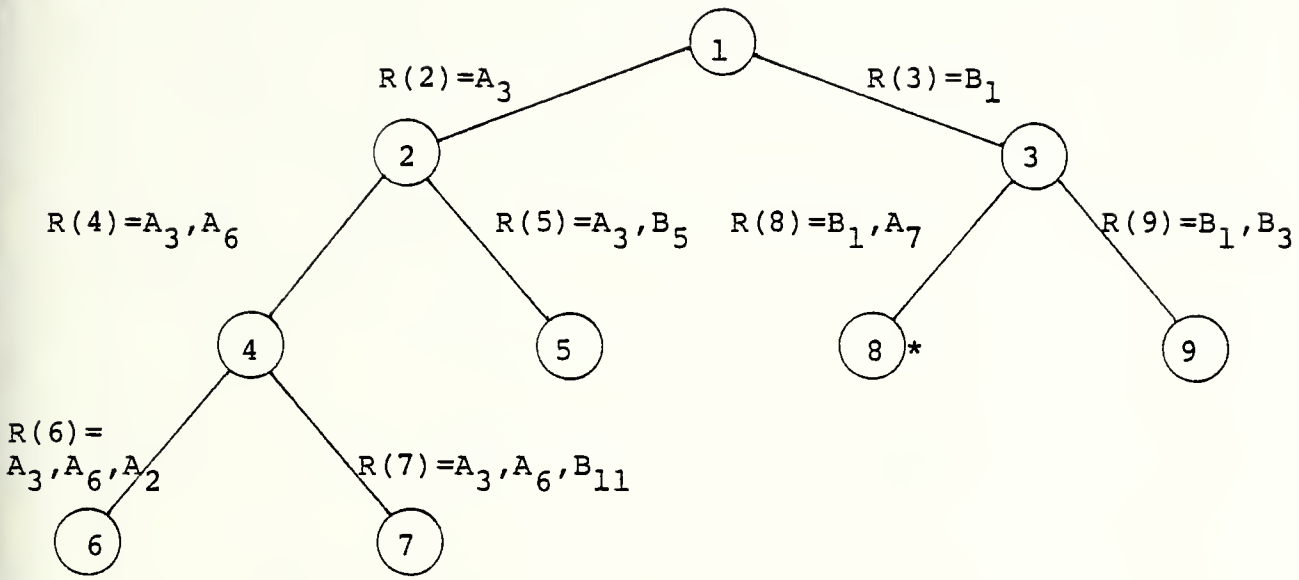
branching, a comparison of terminal node upper bounds and a check for feasibility is performed. When nodes eight and nine branch from node three and the bound of each is computed, it is found that node eight has the highest upper bound of terminal nodes five, six, seven, eight, and nine. It is further found that PA(8) and PB(8) have no targets in common. Therefore, the algorithm stops and PA(8) and PB(8) form the optimal solution with a value of 26.

Table II summarizes the progress of the algorithm in the example. Figure 12 illustrates the complete branching tree for the problem, and Figure 13 shows the optimal path of each aircraft through the target area.

TABLE II

TEST PROBLEM SUMMARY INFORMATION

NODE	PA(I)	V[PA(I)]	PB(I)	V[PB(I)]	$X_{AB}(I)$	VINT(I)	UB(I)
1	A_{13}, A_{11}, A_7, A_3	13	$B_{13}, B_{11}, B_9, B_5, B_1$	16	$A_3 = B_1$	5	29
2	A_{13}, A_{11}, A_7, A_6	13	$B_{13}, B_{11}, B_9, B_5, B_1$	16	$A_6 = B_5$	5	29
3	A_{13}, A_{11}, A_7, A_3	13	$B_{13}, B_{11}, B_9, B_5, B_3$	14	$A_7 = B_3$	3	27
4	$A_{13}, A_{11}, A_9, A_5, A_4, A_2$	12	$B_{13}, B_{11}, B_9, B_5, B_1$	16	$A_2 = B_{11}$	.4	28
5	A_{13}, A_{11}, A_7, A_6	13	B_{13}, B_{11}, B_9, B_1	11	$\emptyset$	0	24
6	A_{13}, A_{11}, A_9, A_1	9	$B_{13}, B_{11}, B_9, B_5, B_1$	16	$A_1 = B_{13}$	2	25
7	$A_{13}, A_{11}, A_9, A_5, A_4, A_2$	12	$B_{13}, B_{10}, B_9, B_5, B_1$	14	$A_5 = B_{10}$	2	26
* 8	A_{13}, A_{11}, A_9, A_3	12	$B_{13}, B_{11}, B_9, B_5, B_3$	14	$\emptyset$	0	26
9	A_{13}, A_{11}, A_7, A_3	13	B_{13}, B_{11}, B_9, B_5	11	$\emptyset$	0	24



*Optimal

Figure 12. Branching Tree for Test Problem

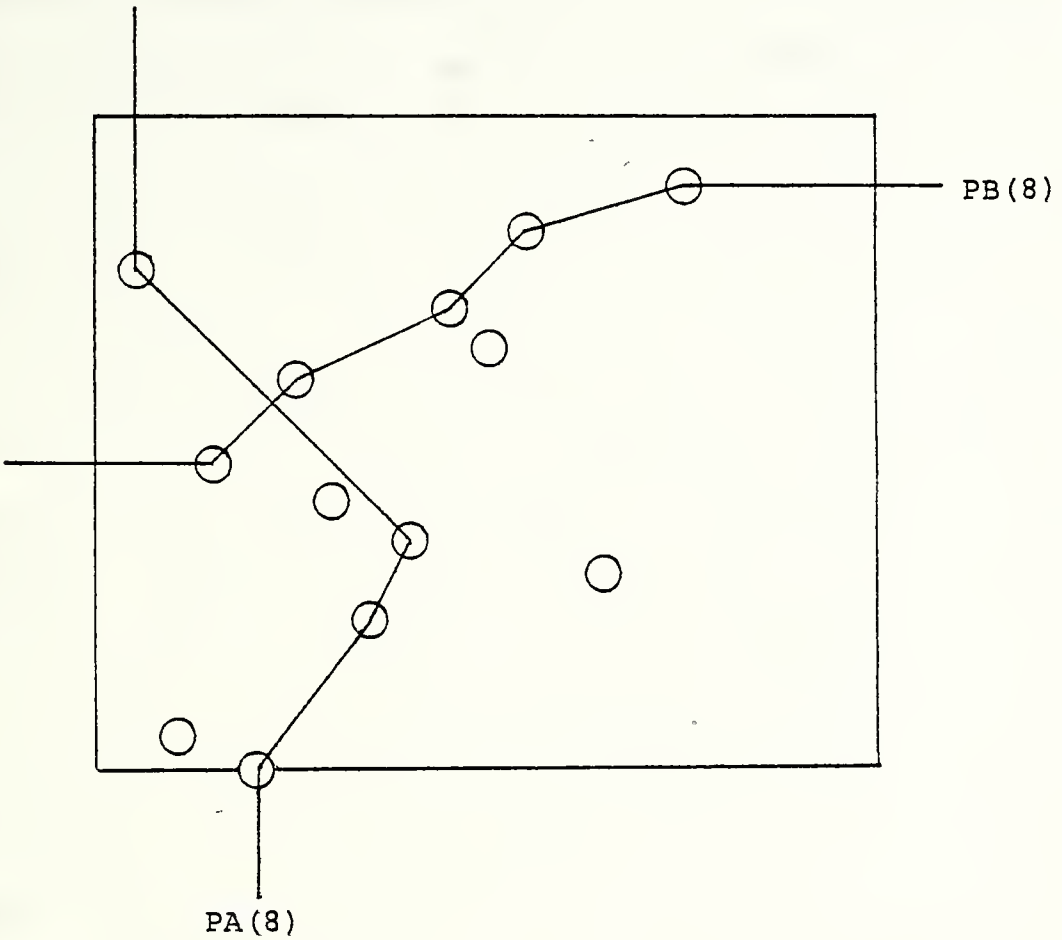


Figure 13. Optimal Path of Each Aircraft

V. VARIATIONS

The model which has been formulated can be modified to solve more difficult problems having additional constraints.

One constraint would be to limit the number of bombs on each aircraft, thus limiting the number of targets allowed in the optimal paths of the aircraft. To incorporate this restriction, the single aircraft D.P. method of Section I must be modified. This can be done by increasing the number of state variables from one to two. At each stage there will be one state variable, x_n, representing the lateral position of the aircraft at stage n. Another state variable, $NBOMBS_n$, denotes the number of bombs remaining in the aircraft at stage n. Although the computation required for the D.P. portion of the algorithm increases exponentially with the number of state variables, practical problems can still be quickly solved with this added constraint. The bomb limitation has been incorporated into the computer program contained in Appendix B.

In a similar manner, a restriction on the total lateral deviation of the aircraft or on the total number of course changes allowed in the target area could be considered.

Another modification would be to have M aircraft attacking from each direction. The single aircraft D.P. portion of the algorithm becomes instead an optimization for a single group of aircraft, where a separate mutually exclusive path

is computed for each aircraft in the group (the group consisting of all aircraft coming from one direction). The optimization for a single group of M aircraft is a dynamic programming problem with M state variables. The lateral position of the i^{th} aircraft at stage n is denoted by x_{in}. The one directional problem is discussed in detail in Ref. 2. To solve the two directional problem, the group of aircraft coming from one direction is viewed as group A, the other as group B. Since there will be no intersection of targets within a group, the only concern will be with targets in common between the two groups. As in Section III, the highest valued common target at node I, $X_{AB}(I)$, can be found and the branching performed with one node of the branch restricting group A to exclude target $X_{AB}(I)_A$, the other node restricting group B to exclude $X_{AB}(I)_B$.

Again, it is critical to realize that doubling the number of state variables far more than doubles the computations required, and eventually the problem will become computationally infeasible.

The next modification to be considered is the problem of aircraft attacking from more than two directions. The general theory of the two-directional problem can be extended to the N directional case, but the rules governing branching become more involved, and the number of nodes required for solution greatly increases. One possible approach where N = 3 will be briefly considered, with the aircraft designated A, B, and C.

At each node, there are two types of intersection possible, a two-aircraft intersection, and a three-aircraft intersection. Branching will be done on the highest valued target of intersection, whether it is common to two aircraft or three. At node I this point will be designated INT(I). Assume that at node L, INT(L) = $A_i = B_j$. A double branch would emanate from node L, one restricting aircraft A to exclude A_i, the other restricting aircraft B to exclude B_j. Should INT(L) instead equal $A_i = B_j = C_k$, a triple branching would be required. One branch would restrict aircraft A and B to exclude A_i and B_j, respectively. The second branch would restrict aircraft A and C to exclude A_i and C_k, respectively. The last branch would restrict aircraft B and C to exclude B_j and C_k, respectively.

The paths at each node are calculated as described in Section I, using the single aircraft D.P. optimization, subject to the restrictions above. The upper bound for any node is the summation of the values of the three paths. The stopping condition is reached, as in Section III, when the terminal node with the highest upper bound has no targets in common on the paths of the three aircraft.

If computer time is critical, a suboptimal solution, as close to optimal as the decision maker desires, could be found. This is done by selecting a value which represents the maximum difference the decision maker can tolerate between the highest upper bound and its corresponding lower bound. When this value is achieved, the suboptimal solution is obtained by randomly

assigning the targets of intersection at that node to either aircraft, thus making the solution feasible.

VI. THE COMPUTER PROGRAM

Appendix B contains a computer program, written in Fortran IV, which will solve the target optimization problem for two aircraft traversing, from different directions, a target area of up to 100 targets. The difference, α, in the initial courses of the two aircraft, may vary between 0 and 360 degrees. The aircraft may have between two and twenty bombs on board. The computer program gives the user the number of branchings required for solution, the optimal value of the targets chosen, the path of targets each aircraft is to attack, and a plot of the target area and the optimal paths through it. Appendix C contains the output from a one hundred target area, with an allowable course deviation of forty-five degrees, six bombs per aircraft, and α = ninety degrees.

The input parameters are the total number of targets, the number of bombs on board each aircraft, the allowable course deviation angle, the difference in the initial courses of the aircraft, and the location and value of each target. The target positions are given in cartesian coordinates, with the X axis perpendicular to the initial course of aircraft A. The angle α is measured counter-clockwise from aircraft A, and is input in degrees. The allowable course deviation angle is identical for both aircraft, although the program could easily be modified to allow each a separate deviation. This angle is also input in degrees.

The program begins by sorting and numbering the targets, first with respect to the initial course of aircraft A, and then with respect to aircraft B.

An MN by MN matrix is formed for determining for either aircraft whether one target position may be feasibly reached from another. A "1" indicates feasibility, and a "0" infeasibility. The diagonal elements from upper right to lower left are all zero, indicating that the aircraft may not remain at one target for more than one stage. Denoting any element as FEAS(I,J), the elements below the diagonal give the feasibility of aircraft A going from target I to target J. The elements above the diagonal give the feasibility of aircraft B going from target J to target I. This matrix eliminates the need to geometrically compute feasibility at every stage of the algorithm.

A vector AB is formed to correlate the target numbers with respect to A with the target numbers with respect to B. If AB(i) = j, then $A_i = B_j$.

The path restrictions for every node are stored in a 100 by 50 matrix R. The matrix permits up to one hundred branchings of the algorithm, and restrictions of up to fifty targets at each node. Either of these may be increased by the user. The restrictions for node I are stored in row I of R by denoting a target A_i as negative i, and a target B_j as j, thus signifying whether a particular numerical element restricts aircraft A or aircraft B.

The single aircraft D.P. portion of the algorithm is performed in subroutine D.P. of the program. A slightly modified version of the double DO loop method suggested in Ref. 2 is used. It is presented in Fortran in a simplified form in Figure 14.

```
      DO 30 I = 2,MN
      DO 20 J = 1,I-1
C     Is it feasible to go from target I to target J?
      DO 10 K = 2,NBOMBS
C     If I have K bombs on board at target I, is J
      the best target to go to?
   10 Continue
   20 Continue
   30 Continue
C     Trace back to find the best path
```

Figure 14. The Simplified Triple DO Loop of Subroutine D.P.

A one-bomb limitation is not allowed, since the solution to the one-bomb problem is merely to choose the two highest valued targets and assign one of them to aircraft A and one to aircraft B. If an unlimited number of targets is possible on a path, as might be the case in planning a photo reconnaissance mission, the bomb limitation should be completely removed from the program, rather than using a very large number for the limitation. This is so because increasing the number of bombs on board increases the computational effort required.

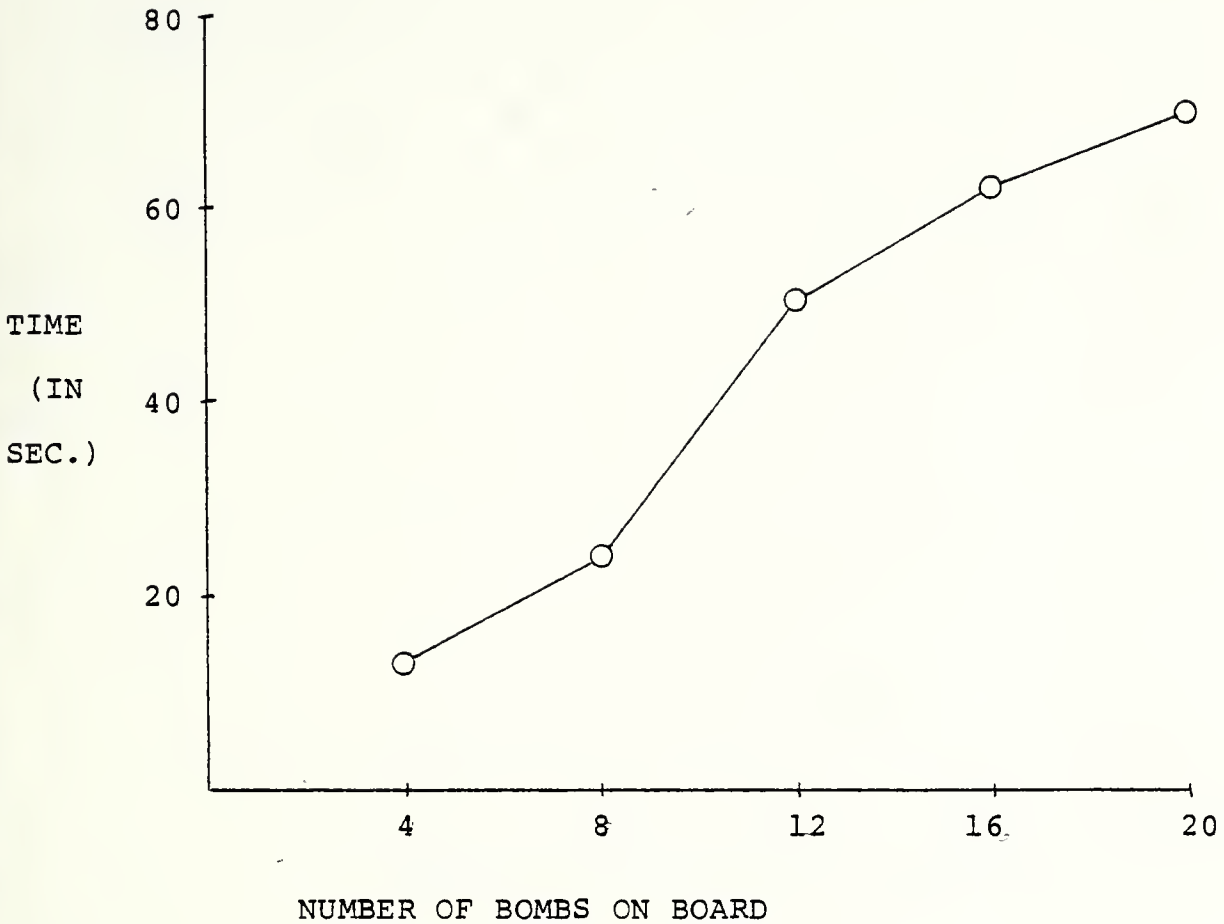
The score ordering method suggested in Ref. 8 was tested in the program. This method requires that for target I, the targets I-1, ..., 1 are stored in a list in order of non-increasing cumulated value. At target I, the list is scanned, starting from the top, until a feasible target J is found. The cumulative value of J is then added to the value of I, and target I is placed in the list, its position depending on its now cumulative value. This eliminates the need to scan all the lower numbered targets from target I to find the best one. It did in fact result in reduced computation for up to three bombs on board. But with more than three, the computation required to update the list at each stage outweighed the savings, and therefore score ordering was not included in the program.

For each node I in the problem, the values of R(I), UB(I), $X_{AB}(I)$, and VINT(I) are saved. PA(I) and PB(I) are discarded as soon as the above four values are computed. Since the branching tree grows horizontally as well as vertically (see Figures 12 and 20), a large amount of storage would be used up in saving the paths. For this reason, new paths are computed at each new node from R(I) and $X_{AB}(I)$ of its predecessor node I.

Figures 15, 16, and 17 are time comparisons for various input values of NBOMBS, course deviation angles, and α. The data points on the graphs represent averages for three different random target areas. However, the trends were almost identical for each set of targets.

Increasing the number of bombs per aircraft causes an almost linear increase in execution time. An increase in the course deviation angle approximates an exponential increase in the time required. Decreasing α causes execution time to increase. This should be expected, since the closer to parallel the two aircraft are, the larger the number of targets in common at each node is likely to be, resulting in more branching being required.

Figure 18 plots execution time versus the number of targets in an area. Figure 19 illustrates the corresponding number of branchings required. As the total number of targets increases, the execution times tend to cycle. One possible explanation for this is that as the number of targets increases, the amount of computation in the dynamic programming subroutine increases. However, when a path intersection occurs, more targets provide more alternative paths which may be feasible and optimal, thereby reducing the number of branchings required. For certain numbers of targets, the time savings from the reduced branchings override the increased subroutine computation required, thus reducing total solution time. Similar trends were found with other target arrangements. This again points out the unpredictability of the computation required for solution.

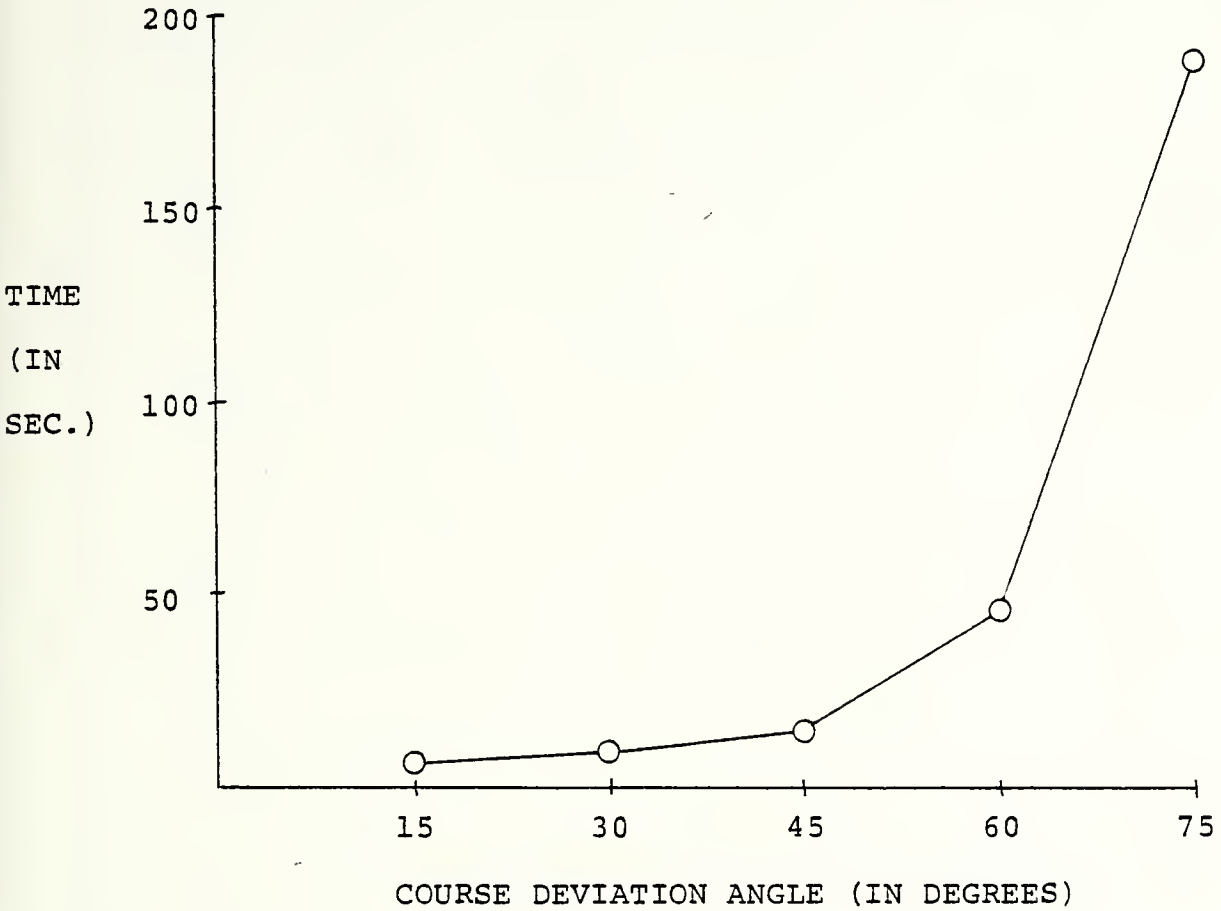


TOTAL NUMBER OF TARGETS = 100

COURSE DEVIATION ANGLE = 45°

α = 90°

Figure 15. Execution Time Versus Bombs Per Aircraft

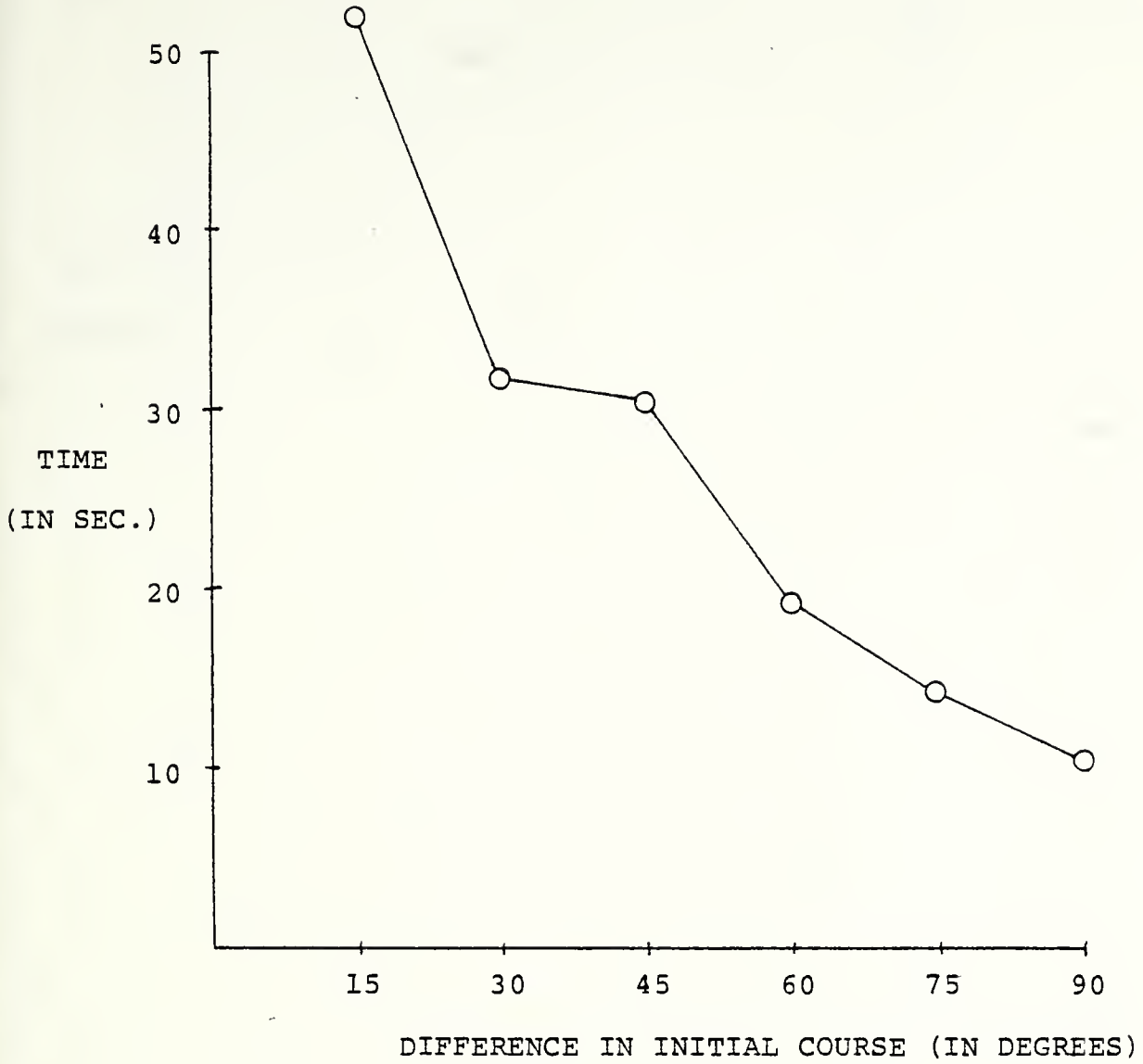


TOTAL NUMBER OF TARGETS = 100

NUMBER OF BOMBS PER AIRCRAFT = 6

$\alpha = 90°$

Figure 16. Execution Time Versus Course Deviation Angle

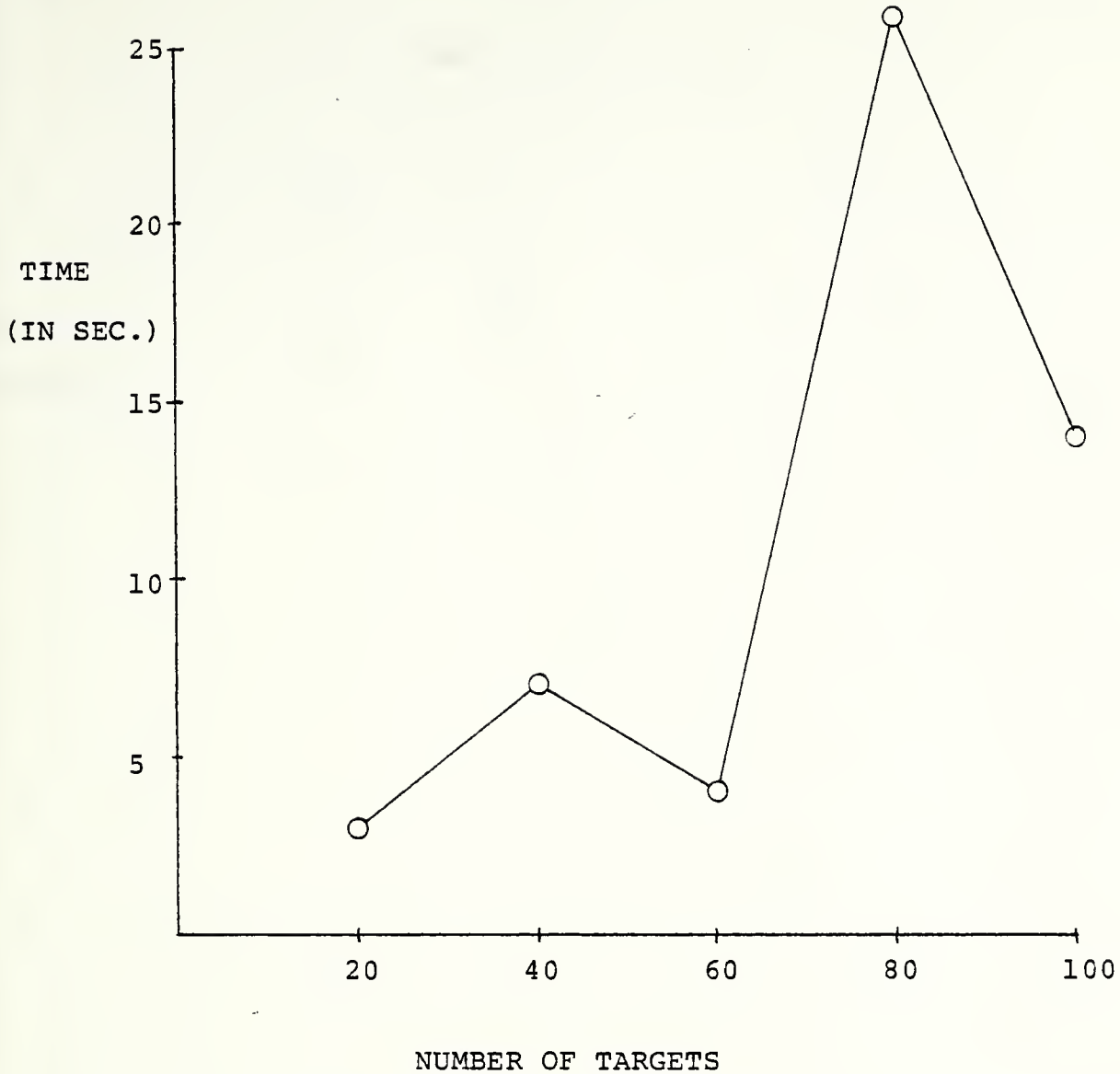


TOTAL NUMBER OF TARGETS = 100

NUMBER OF BOMBS PER AIRCRAFT = 6

COURSE DEVIATION ANGLE = 45°

Figure 17. Execution Time Versus α

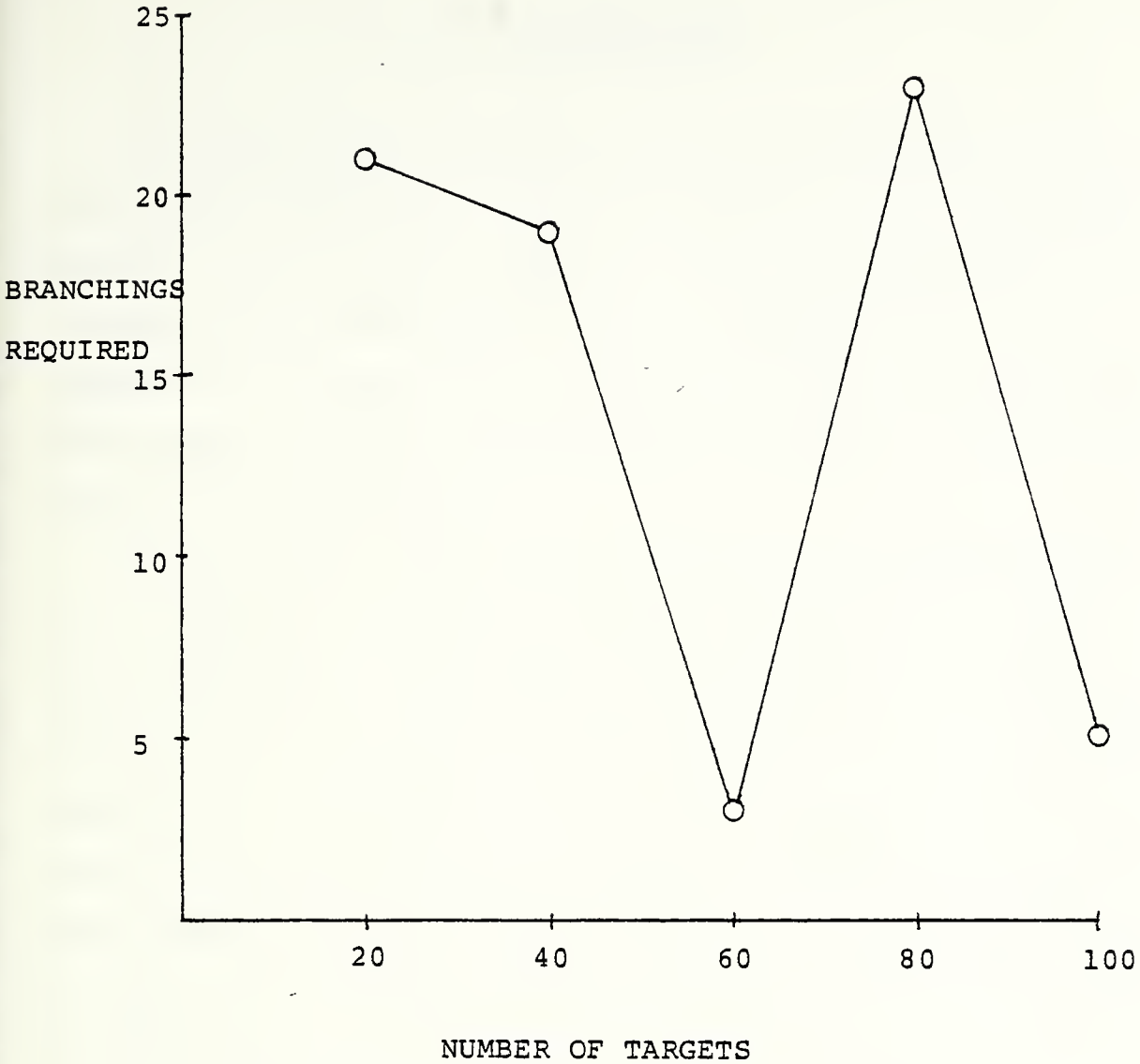


NUMBER OF BOMBS PER AIRCRAFT = 6

COURSE DEVIATION ANGLE = 45°

α = 90°

Figure 18. Execution Time Versus Number of Targets



NUMBER OF BOMBS PER AIRCRAFT = 6

COURSE DEVIATION ANGLE = 45°

α = 90°

Figure 19. Required Branchings Versus Number of Targets

VII. CONCLUSIONS

The algorithm and computer program presented can solve the two-aircraft target optimization problem for up to one hundred targets, with an aircraft payload of up to twenty bombs, using a minimal amount of computer time. In addition, it can be expanded by the user to suit his specific needs, including more targets, larger payloads, more aircraft, and a multiple direction attack scenario.

The algorithm itself could be improved if there were some way to recognize the optimal solution before the stopping condition was met. Oftentimes, an early node will produce the optimal path, but the algorithm continues, because the node's upper bound is not the highest. This is the case where multiple optimal solutions exist, and each level of the branching tree produces equivalent upper bounds. This is illustrated in Figure 20. The upper bounds are indicated above each node. At node 13, an upper bound of thirty-seven is achieved with no targets in common for PA(13) and PB(13). Yet the branching must continue, since other terminal nodes have higher upper bounds. Finally, by node 25, it is realized that the solution found at node 13 was in fact optimal. Had this been realized at node 13, the computation required could have been cut in half.

The computer program was developed to test the algorithm, and should not be considered as an end-product software package.

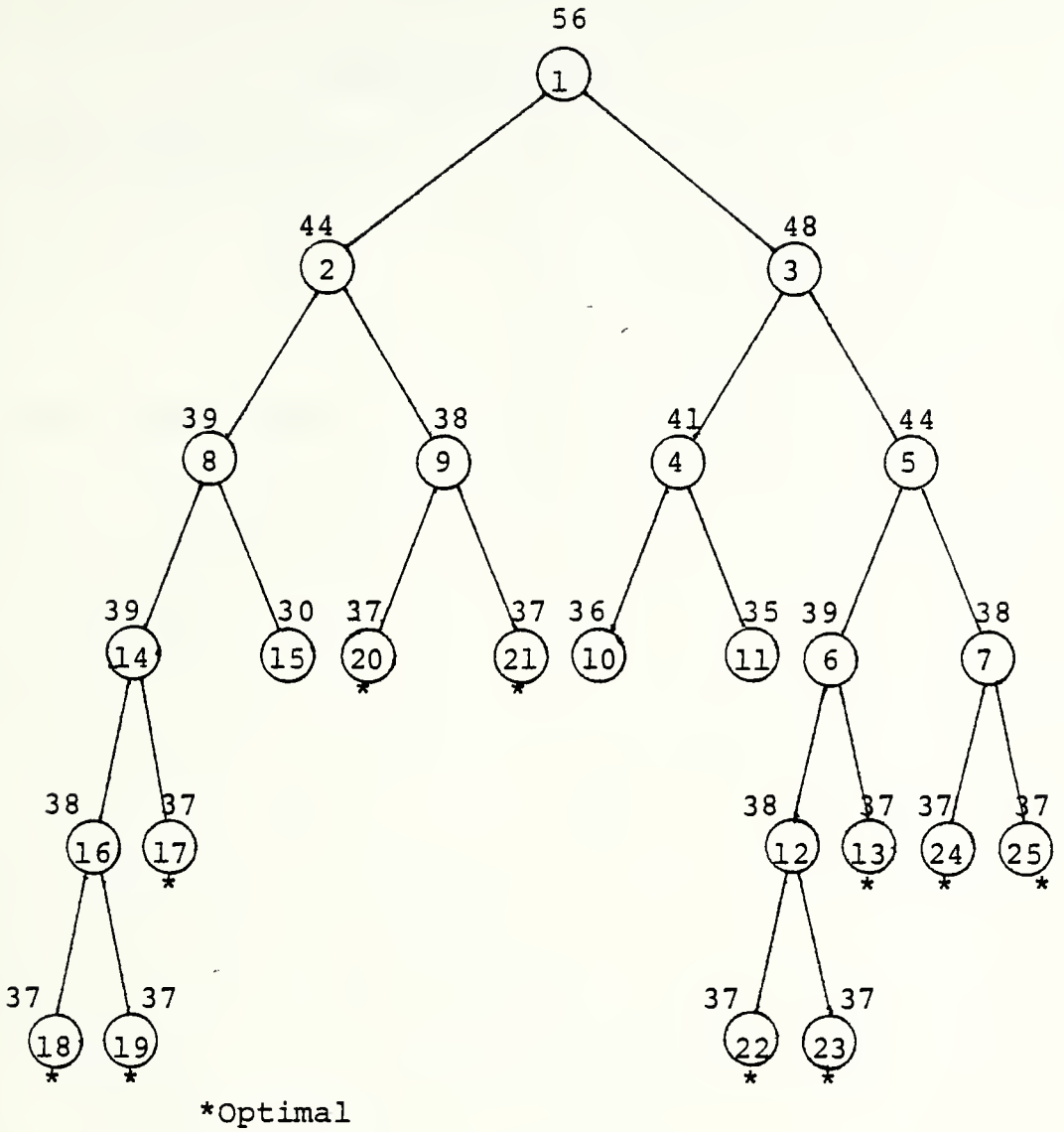


Figure 20. A Branching Tree with Alternative Optimal Solutions.

It is storage inefficient, especially in the area of the R matrix and the fact that information is kept in storage for all nodes rather than just the terminal ones. Improvements in these areas could be implemented if desired.

APPENDIX A

GLOSSARY OF TERMS

Allowable Course Deviation = the allowable number of degrees that an aircraft is permitted to deviate from its initial course upon entering the target area.

MN = the total number of targets in the target area.

A_i = the i^{th} target, ordered non-increasingly by its parallel distance from the boundary at which aircraft A enters the target area, i = 1, ..., MN.

B_j = the j^{th} target, ordered non-increasingly by its parallel distance from the boundary at which aircraft B enters the target area, j = 1, ..., MN.

$V[A_i]$ = the value of target A_i.

$V[B_j]$ = the value of target B_j.

PA(I) = the set or path of targets for aircraft A computed at node I. It is feasible and optimal for the single aircraft problem.

PB(I) = the set or path of targets for aircraft B computed at node I. It is feasible and optimal for the single aircraft problem.

V[PA(I)] = the summation of the values of the targets in PA(I).

V[PB(I)] = the summation of the values of the targets in PB(I).

$X_{AB}(I)$ = the target common to both PA(I) and PB(I) which has the highest value.

$X_{AB}(I)_A$ = $X_{AB}(I)$ in terms of aircraft A.

$X_{AB}(I)_B$ = $X_{AB}(I)$ in terms of aircraft B.

VINT(I) = the summation of the values of all targets common to both PA(I) and PB(I).

UB(I) = V[PA(I)] + V[PB(I)] = an upper bound on the optimal solution at node I.

LB(I) = UB(I) - VINT(I) = a lower bound on the optimal solution at node I.

R(I) = a set of restrictions on PA(I) and PB(I) consisting of a listing of targets which each must exclude.

NBOMBS = the number of bombs with which an aircraft enters the target area.

α = the difference in the initial courses of the two aircraft, measured in degrees counter-clockwise from aircraft A.

APPENDIX B

THE COMPUTER CODE

```
C A TWO DIRECTIONAL TARGET OPTIMIZATION MODEL USING THE
C BRANCH AND BOUND ALGORITHM WITH DYNAMIC PROGRAMMING.
C
C
C THIS PROGRAM COMPUTES OPTIMAL TARGET PATHS FOR TWO
C AIRCRAFT TRAVERSING A TARGET AREA FROM DIFFERENT
C DIRECTIONS. IT WILL PERFORM UP TO 100 BRANCHINGS. THE
C TARGET POSITIONS ARE IN TERMS OF THE CARTESIAN PLANE, WITH
C THE X AXIS PERPENDICULAR TO ONE OF THE AIRCRAFT'S INITIAL
C COURSES. THE NUMBER OF BOMBS ON BOARD EACH AIRCRAFT MUST
C BE BETWEEN TWO AND TWENTY. THE TARGET AREA MAY HAVE UP TO
C ONE HUNDRED TARGETS.
C
C THE INPUT PARAMETERS ARE :
C  MN = TOTAL NUMBER OF TARGETS
C  NBOMBS = NUMBER OF BOMBS ON BOARD EACH AIRCRAFT
C  ANGLE = ALLOWABLE COURSE DEVIATION ( IN DEGREES )
C  ALPHA = DIFFERENCE IN INITIAL COURSES OF THE TWO AIRCRAFT
C  AX(I) = X COORDINATE OF TARGET I
C  AY(I) = Y COORDINATE OF TARGET I
C  VA(I) = VALUE OF TARGET I
C
C
C MAIN PROGRAM
C
C
C DIMENSION THE ARRAYS AND ASSIGN VARIABLE TYPE
      REAL AX(100),AY(100),VA(100),XA(100),YA(100),BX(100),
     1BY(100),VB(100),XB(100),YB(100),TVA(100),TVB(100),
     1VINT(100),UB(100),V(100)
      INTEGER FEAS(100,100),R(100,50),AB(100),XAB(100),
     1PATH(100),PA(100)
      INTEGER COUNT
C
C
C READ IN THE NUMBER OF TARGETS, THE NUMBER OF BOMBS PER
C AIRCRAFT, THE DEVIATION ANGLE, AND THE DIFFERENCE IN
C INITIAL COURSES OF THE TWO AIRCRAFT.
      READ (5,10) MN,NBOMBS,ANGLE,ALPHA
   10 FORMAT (2I4,2F5.0)
C
C
C READ IN THE TARGET POSITIONS AND VALUES
      DO 20 I=1,MN
      READ (5,30) AX(I),AY(I),VA(I)
   20 CONTINUE
   30 FORMAT (3F12.0)
```

```
C
C
C  SORT TARGETS FOR AIRCRAFT A
      MNM1 = MN-1
      DO 50 I=1,MNM1
      IP1 = I+1
      IOPT = 0
      DO 40 J=IP1,MN
      IF (AY(I).GE.AY(J)) GO TO 40
      TEMPY = AY(I)
      TEMPX = AX(I)
      TEMPV = VA(I)
      AY(I) = AY(J)
      AX(I) = AX(J)
      VA(I) = VA(J)
      AY(J) = TEMPY
      AX(J) = TEMPX
      VA(J) = TEMPV
   40 CONTINUE
   50 CONTINUE
C
C
C  SORT TARGETS FOR AIRCRAFT B
      ALPHA = ALPHA*3.141592654/180.0
      DO 60 I=1,MN
      BX(I) = AX(I)*COS(ALPHA)+AY(I)*SIN(ALPHA)
      BY(I) = -AX(I)*SIN(ALPHA)+AY(I)*COS(ALPHA)
      VB(I) = VA(I)
   60 CONTINUE
      DO 80 I=1,MNM1
      IP1 = I+1
      DO 70 J=IP1,MN
      IF (BY(I).GE.BY(J)) GO TO 70
      TEMPY = BY(I)
      TEMPX = BX(I)
      TEMPV = VB(I)
      BY(I) = BY(J)
      BX(I) = BX(J)
      VB(I) = VB(J)
      BY(J) = TEMPY
      BX(J) = TEMPX
      VB(J) = TEMPV
   70 CONTINUE
   80 CONTINUE
C
C
C  COMPUTE FEASIBILITY MATRIX
      ANGLE = ((90.0-ANGLE)*3.141592654/180.0)-1.0E-8
      FEAS(1,1) = 0
      DO 120 I=2,MN
      IM1 = I-1
      FEAS(I,I) = 0
      DO 110 J=1,IM1
      IF (AX(I).EQ.AX(J)) GO TO 90
      IF (BX(I).EQ.BX(J)) GO TO 100
      SLOPEA = ATAN(ABS((AY(J)-AY(I))/(AX(J)-AX(I))))
      SLOPEB = ATAN(ABS((BY(J)-BY(I))/(BX(J)-BX(I))))
      FEAS(I,J) = 0
      FEAS(J,I) = 0
      IF (SLOPEA.GT.ANGLE) FEAS(I,J)=1
      IF (SLOPEB.GT.ANGLE) FEAS(J,I)=1
      GO TO 110
   90 FEAS(I,J) = 1
      GO TO 110
  100 FEAS(J,I) = 1
  110 CONTINUE
  120 CONTINUE
C
C
```

```
C SET UP AB CORRELATION VECTOR
      DO 140 L=1,MN
      DO 130 M=1,MN
      TWOPIE = 2.0*3.141592654
      TPMA = TWOPIE-ALPHA
      IF (ABS(AX(L)-(BX(M)*COS(TPMA)+BY(M)*SIN(TPMA))).GT.
     11.0E-2) GO TO 130
      IF (ABS(AY(L)-(-BX(M)*SIN(TPMA)+BY(M)*COS(TPMA))).GT.
     11.0E-2) GO TO 130
      AB(L) = M
      GO TO 140
  130 CONTINUE
  140 CONTINUE
C
CC
C BEGIN BRANCH AND BOUND
CC
C ZERO OUT RESTRICTION MATRIX
      DO 160 L=1,100
      DO 150 M=1,50
      R(L,M) = 0
  150 CONTINUE
  160 CONTINUE
C
CC
C DETERMINE R VECTOR FOR NODE I
      XAB(1) = 0
      I = 0
      J = 1
      K = 0
  170 I = I+1
      IF (I.EQ.101) GO TO 490
      DO 180 L=1,50
      LAST = L
      IF (R(J,L).EQ.0) GO TO 190
      IF (L.EQ.50) GO TO 510
      R(I,L) = R(J,L)
  180 CONTINUE
  190 IF (I.EQ.2) K = 0
      IF (K.EQ.1) GO TO 200
      R(I,LAST) = -XAB(J)
      K = 1
      GO TO 210
  200 R(I,LAST) = AB(XAB(J))
      K = 0
C
CC
C ADJUST TARGET VALUES TO EXCLUDE TARGETS IN R(I)
  210 DO 230 L=1,50
      IF (R(I,L).EQ.0) GO TO 240
      IF (R(I,L).GT.0) GO TO 220
      TVA(-R(I,L)) = VA(-R(I,L))
      VA(-R(I,L)) = -10000.0
      GO TO 230
  220 TVB(R(I,L)) = VB(R(I,L))
      VB(R(I,L)) = -10000.0
  230 CONTINUE
C
CC
C CALL DP FOR A AND B
  240 ICHK = 1
      CALL DP (FEAS,VA,ICHK,MN,NBOMBS,PATH,W,COUNT)
      VPA = W
      NPA = COUNT
      DO 250 L=1,NPA
      PA(L) = PATH(L)
  250 CONTINUE
      ICHK = 2
      CALL DP (FEAS,VB,ICHK,MN,NBOMBS,PATH,W,COUNT)
      IF (IOPT.EQ.1) GO TO 350
C
```

```
C
C RESET TARGET VALUES
      DO 270 L=1,50
      IF (R(I,L).EQ.0) GO TO 280
      IF (R(I,L).GT.0) GO TO 260
      VA(-R(I,L)) = TVA(-R(I,L))
      GO TO 270
  260 VB(R(I,L)) = TVB(R(I,L))
  270 CONTINUE
C
C
C COMPUTE XAB(I), VINT(I), AND UB(I)
  280 XMAX = 0.0
      SUM = 0.0
      DO 300 L=1,NPA
      DO 290 M=1,COUNT
      IF (AB(PA(L)).NE.PATH(M)) GO TO 290
      SUM = SUM+VA(PA(L))
      IF (VA(PA(L)).LE.XMAX) GO TO 300
      XMAX = VA(PA(L))
      XAB(I) = PA(L)
      GO TO 300
  290 CONTINUE
  300 CONTINUE
      VINT(I) = SUM
      UB(I) = VPA+W
      IF (I.EQ.1.AND.VINT(I).EQ.0.0) GO TO 340
      IF (K.EQ.1) GO TO 170
      UB(J) = 0.0
C
C
C CHOOSE HIGHEST UPPER BOUND OF ALL TERMINAL NODES
C AND CHECK IF IT IS FEASIBLE
      UBMAX = 0.0
      DO 310 L=1,I
      IF (UB(L).LE.UBMAX) GO TO 310
      UBMAX = UB(L)
      J = L
  310 CONTINUE
      IF (VINT(J).EQ.0.0) GO TO 330
C
C
C CHECK FOR TIE FOR HIGHEST UPPER BOUND.
C IF THERE IS A TIE, IS IT FEASIBLE?
      DO 320 L=1,I
      IF (UB(L).NE.UB(J)) GO TO 320
      IF (VINT(L).NE.0.0) GO TO 320
      J = L
      GO TO 330
  320 CONTINUE
      GO TO 170
C
C
C IF NECESSARY, RECOMPUTE OPTIMAL PATH
  330 NODES = I
      IF (I.EQ.J) GO TO 350
      I = J
      IOPT = 1
      GO TO 210
C
C
```

```
C PRINT OUT THE NUMBER OF BRANCHING NODES, THE OPTIMAL
C VALUE, AND THE OPTIMAL PATHS.
  340 NODES = I
  350 WRITE (6,360)
  360 FORMAT ('1')
      WRITE (6,370) NODES
  370 FORMAT (' OPTIMAL SOLUTION FOUND IN',I3,' BRANCHINGS',
     1////////)
      WRITE (6,380) UB(J)
  380 FORMAT (' THE VALUE OF THE OPTIMAL SOLUTION IS ',F12.4
     1,////////)
      WRITE (6,390)
  390 FORMAT (' OPTIMAL AIRCRAFT PATHS',///)
      WRITE (6,400)
  400 FORMAT (6X,'AIRCRAFT  A',///)
      WRITE (6,410)
  410 FORMAT (4X,'X',12X,'Y',//)
      DO 420 I=1,NPA
      XA(I) = AX(PA(I))
      YA(I) = AY(PA(I))
      WRITE (6,430) XA(I),YA(I)
  420 CONTINUE
  430 FORMAT (1X,F8.2,5X,F8.2,/)
      WRITE (6,440)
  440 FORMAT (///,6X,'AIRCRAFT  B',///)
      WRITE (6,410)
      DO 450 I=1,COUNT
      XB(I) = BX(PATH(I))*COS(TPMA)+BY(PATH(I))*SIN(TPMA)
      YB(I) = -BX(PATH(I))*SIN(TPMA)+BY(PATH(I))*COS(TPMA)
      WRITE (6,430) XB(I),YB(I)
  450 CONTINUE
C
C
C PLOT THE TARGETS AND THE PATHS
      WRITE (6,460)
  460 FORMAT ('1')
      CALL PLOTP (AX,AY,MN,1)
      CALL PLOTP (XA,YA,NPA,2)
      CALL PLOTP (XB,YB,COUNT,3)
      WRITE (6,470)
  470 FORMAT (///,40X,'PATH OF AIRCRAFT A DENOTED BY +')
      WRITE (6,480)
  480 FORMAT (/,40X,'PATH OF AIRCRAFT B DENOTED BY *',
     1//////////)
      GO TO 530
C
C
C CHECK FOR DIMENSIONAL ERRORS
  490 WRITE (6,500)
  500 FORMAT (////,' REQUIRED BRANCHING EXCEEDS 100 NODES',
     1////)
      GO TO 530
  510 WRITE (6,520)
  520 FORMAT (////,' RESTRICTION VECTOR EXCEEDS 50',////)
  530 STOP
      END
```

```
      SUBROUTINE DP (FEAS,V,ICHK,MN,NBOMBS,PATH,W,COUNT)
C
C THIS SUBROUTINE COMPUTES THE OPTIMAL PATH FOR A SINGLE
C AIRCRAFT TRAVERSING THE TARGET AREA.
C
C
      REAL VV(100,20),V(100),CVAL(20)
      INTEGER P(100,20),FEAS(100,100),MOVE(20),PATH(100)
      INTEGER COUNT
      CVAL(1) = 0.0
      MOVE(1) = 200
      DO 20 K=1,NBOMBS
      VV(1,K) = V(1)
      P(1,K) = 200
   20 CONTINUE
C
C
C I IS THE TARGET THAT THE AIRCRAFT IS AT PRESENTLY
      DO 70 I=2,MN
      VV(I,1) = V(I)
      DO 30 K=2,NBOMBS
      CVAL(K) = 0.0
      MOVE(K) = 200
      VV(I,K) = V(I)
   30 CONTINUE
      IM1 = I-1
C
C
C J IS THE TARGET THAT THE AIRCRAFT MAY GO TO FROM I
      DO 50 J=1,IM1
      IF (ICHK.EQ.1.AND.FEAS(I,J).EQ.0) GO TO 50
      IF (ICHK.EQ.2.AND.FEAS(J,I).EQ.0) GO TO 50
C
C
C K IS THE NUMBER OF BOMBS LEFT AT TARGET I
      DO 40 K=2,NBOMBS
      IF ((VV(I,K)+VV(J,(K-1))).LE.CVAL(K)) GO TO 40
      CVAL(K) = VV(I,K)+VV(J,(K-1))
      MOVE(K) = J
   40 CONTINUE
   50 CONTINUE
      DO 60 K=1,NBOMBS
      IF (CVAL(K).NE.0.0) VV(I,K)=CVAL(K)
      P(I,K) = MOVE(K)
   60 CONTINUE
   70 CONTINUE
C
C
C COMPUTE FIRST TARGET
      CVAL(NBOMBS) = 0.0
      DO 80 J=1,MN
      IF (VV(J,NBOMBS).LE.CVAL(NBOMBS)) GO TO 80
      CVAL(NBOMBS) = VV(J,NBOMBS)
      MOVE(NBOMBS) = J
   80 CONTINUE
      W = CVAL(NBOMBS)
C
C
C COMPUTE OPTIMAL PATH
      PATH(1) = MOVE(NBOMBS)
      COUNT = 1
      LEFT = NBOMBS+1
      DO 90 I=2,MN
      COUNT = I
      LEFT = LEFT-1
      PATH(I) = P((PATH(I-1)),LEFT)
      IF (P((PATH(I)),(LEFT-1)).EQ.200) RETURN
   90 CONTINUE
      RETURN
      END
```

APPENDIX C

THE COMPUTER OUTPUT

OPTIMAL SOLUTION FOUND IN 7 BRANCHINGS

THE VALUE OF THE OPTIMAL SOLUTION IS 216.2231

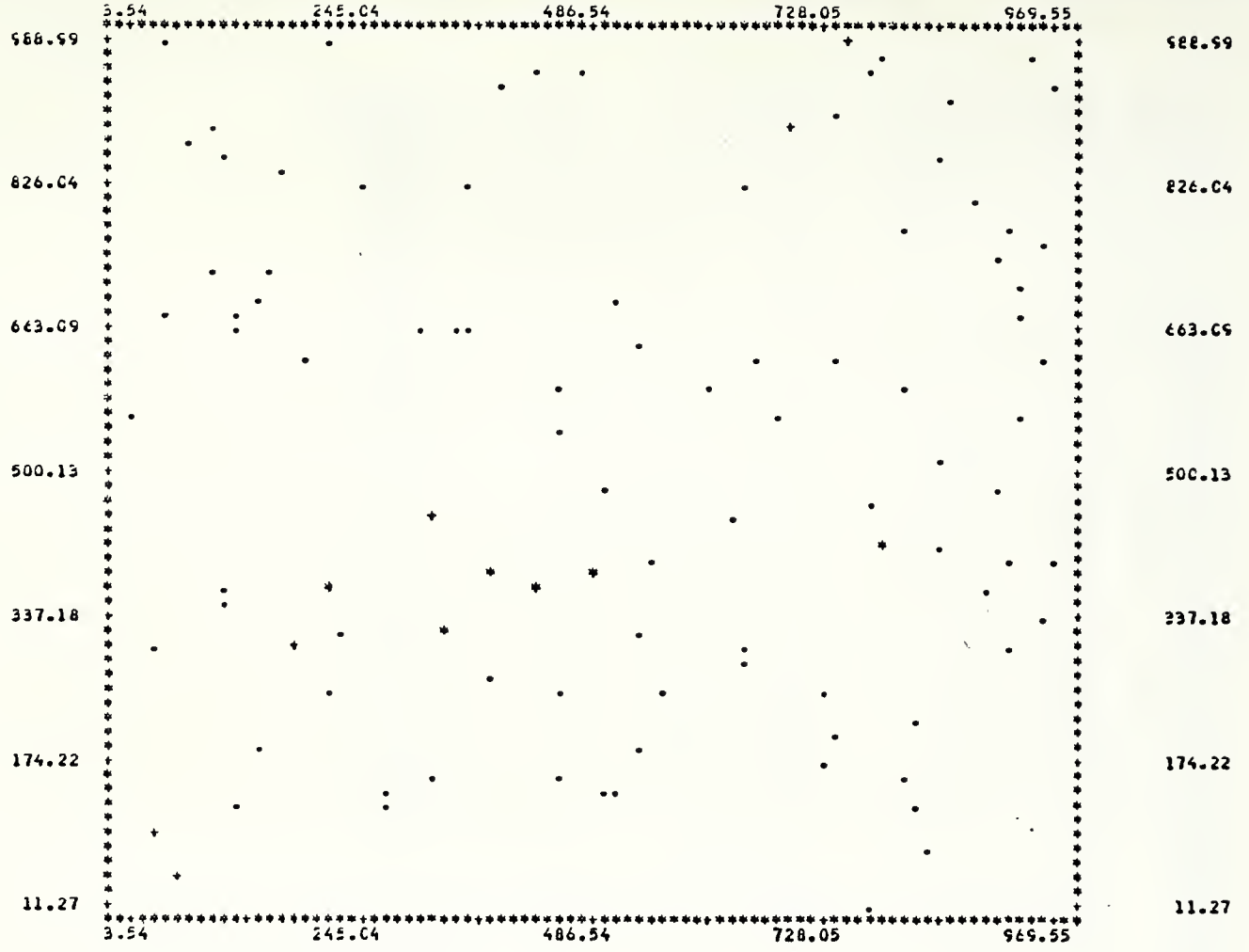
OPTIMAL AIRCRAFT PATHS

AIRCRAFT A

X	Y
47.71	48.91
31.71	99.13
171.81	297.32
317.05	445.82
689.03	886.22
754.96	985.23

AIRCRAFT B

X	Y
794.25	411.59
481.19	389.44
424.49	362.81
383.86	378.05
328.99	324.02
209.62	363.97



X-SCALE: "*"= C.121E 02 UNITS

Y-SCALE: "*"= C.163E 02 UNITS

PATH OF AIRCRAFT A DENOTED BY +

PATH OF AIRCRAFT B DENOTED BY *

BIBLIOGRAPHY

1. Howard, G.T., Minimizing the Number of Penetrations in a Boundary Defense Problem, Naval Postgraduate School Publication Number 55HK72041A, April 1972.

2. Howard, G.T., A Target Selection Model, Naval Postgraduate School Publication Number 55-78-30, October 1978.

3. Nemhauser, G.L., Introduction to Dynamic Programming, pp. 14-84, Wiley, 1966.

4. Garfinkel, R.S. and Nemhauser, G.L., Integer Programming, pp. 108-153, Wiley, 1972.

5. Little, J.D.C., Murty, K.G., Sweeney, D.W., and Karel, C., "An Algorithm for the Traveling Salesman Problem," Operations Research, V. 11, pp. 972-989, November-December 1963.

6. Mitten, L.G., "Branch and Bound Methods: General Formulation and Properties," Operations Research, V. 18, pp. 24-34, January-February 1970.

7. Lawler, E.L. and Wood, D.E., "Branch and Bound Methods: A Survey," Operations Research, V. 14, pp. 699-719, July-August 1966.

8. Martin, C.F. and Poling, R.S., Fast Dynamic Programming Selection Algorithms for the Job Shop Problem with Job Availability Intervals, paper presented at the Western Section ORSA meeting, San Diego, California, September 1977.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	Professor G. T. Howard, Code 55HK Department of Operations Research Naval Postgraduate School Monterey, California 93940	3
5.	Professor J. K. Hartman, Code 55HH Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
6.	LT Gregory R. Hamelin 25 Chestnut Road Verona, New Jersey 07044	3

Thesis
H1626
c.1
Hamelin
181599
A two-directional target optimization model.